When making interactive visualizations, using web–based tools is a very reasonable choice. When working with the programming interface of web-pages, the Document Object Model (*DOM*), the standard is to use a few different languages. To define the content of a page, Hyper Text Markup Language (*HTML*) is used. *HTML* by itself does not provide many options regarding styling, this is where Cascading Style Sheets (*CSS*) comes in. Using *CSS*, *DOM* elements can be styled to one's desire. However, even then, there is still two quite important components for effective visualization missing. Firstly, *HTML* is mostly made for handling text and images, not generating shapes. This is where Scalable Vector Graphics (*SVG*) comes along. *SVG* has some relatively simple shapes built in, such as rectangles and circles, as well as the powerful `path` tag. Using this tag, very sophisticated shapes can be made. Being a vector graphics format, *SVG* will have sharp edges, and generally look as intended no matter how far it is scaled.
Even with all that, there is still no interactivity. This is where *JavaScript* comes along. Being a scripting language for the *DOM*, *JavaScript* is a very powerful tool to have, as it provides runtime modification of the *DOM*.
JavaScript has the ability to listen for events on *DOM* elements, such as mouse clicks, which is what allows dynamically changing visualizations based on user interaction.
Now, just because we have these languages available, does not mean that creating visualizations will be child's play. With plain *JavaScript*, going from data to interactive visualizations would be quite a journey.
This is where the *JavaScript* library "Data-Driven Documents" (*D3*) turns out to be a powerful and versatile choice.
Before *D3*, the people behind it worked on a visualization framework under the name "Protovis" [**?**]. Built with the focus of a user being able to pump out visualizations without much effort, the framework wasn't built for letting the user work directly with the standards mentioned earlier. This lead to slow adoption of the framework, and inefficient use. This is described more in-depth in the article "D3: Data-Driven Documents"[**?**].

The resulting *D3* provides powerful tools for binding data to the *DOM*, while leaving the creation of visualizations to the user of the library. This way the user can utilize the synergy between *HTML*, *SVG*, *CSS* and *JavaScript*.

While this way of doing visualizations is preferred in a lot of cases due to reusability, and user interactivity, working with large datasets can give unwanted results. If the user has to load more than a few megabytes of data in order to create the visualization, it is going to take some time, and if that data result in a large amount of SVG elements, a dynamic or interactive visualization could become unresponsive, which is undesired, as it will lower user experience.