Lennart Buchmann, Martin Bingel, Alaa Aldin Karkoutli, Jonas Gwozdz, Linus Herterich

Technologierecherche

Sammlung vorhandener webbasierter OpenSource Projekte zum erstellen / editieren von Graphen

mermaid.js

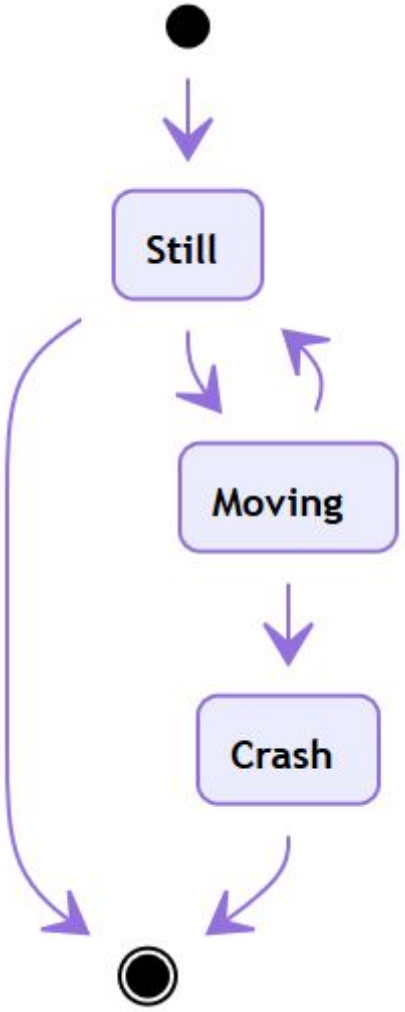
https://mermaidjs.github.io/#/
Demo: https://mermaidjs.github.io/#/stateDiagram

-Mermaid.js generiert Graphs, Diagramme usw. mithilfe des JavaScripts.
-Er bietet verschiedene Typen von Diagrammen und Graphen an. Z.B.:

- Sequence-Diagramm: ist ein Diagramm, bei dem folgendes gezeigt wird, wie Prozesse in welcher Reihenfolge ablaufen.

- Flowchart: Bei dem werden Graphen in verschiedenen Formen und ihre Richtungen deklariert werden, sowie Styling von Knoten (Graphen) und Kanten (Richtungen).

- State-Diagramm: Bei dem werden die Systeme durch Zustände beschrieben werden, wobei die Zustände sich durch Übergänge (die Wege zwischen den Zuständen) verändert werden können.



. u.v.m.

-Dokus sind selbstverständlich verfügbar und ausführlich, also alles ist dokumentiert, was als ein größerer Vorteil gilt.

-Es ist ziemlich einfach Mermaid.js zu installieren. Es gibt sogar einen Live-Editor zum Testen der ganzen Arbeit und sie als .svg-Datei zu exportieren.

-Einarbeitungsdauer kann man nicht genau einschätzen. Also die JavaScript Vorkenntnisse sind von Vorteil, damit Einarbeitungsdauer gekürzt werden könnte. Ansonsten ist es leicht die Funktionsweise von Mermaid.js kapieren zu können.

```
mermaidAPI.initialize({
    startOnLoad:true
});
$(function(){
    const graphDefinition = 'graph TB\na-->b';
    const cb = function(svgGraph){
        console.log(svgGraph);
    };
    mermaidAPI.render('id1',graphDefinition,cb);
});
```

Rete.js

https://github.com/retejs/rete
Demo: https://codepen.io/Ni55aN/pen/xzgQYq

- Rete ist ein Framework zum ersten webbasierte Graphen-editor (Knoten und Kanten)
- Installation/Einbindung in drei Formaten (UMD (*.min.js), CommonJS (*.common.js), ES модули (*.esm.js))
- Import/Export in JSON (andere Datentypen durch Plugins)
- vergleichsweise spärliche Dokumentation
- vergleichsweise geringe Community (3.6k)
- MIT License (Open-Source)
- Event-based Architektur (neue Funktionen in Form von Plugins)
- somit beliebig erweiterbares Framework
- Anmerkung: Ich würde von diesem Framework abraten da es vergleichsweise knapp Dokumentiert ist. Mir fiel es schwer Informationen zu sammeln und das ist keine gute Voraussetzung

bpmn-js

http://bpmn.io/toolkit/bpmn-js/walkthrough/
Demo:https://demo.bpmn.io/new

- 2 Methoden der Nutzung:
    - pre-packaged-version
        - einfachere nutzung
    - npm(Notable Peru Mariachis) version (für uns nützlicher)
        - kompliziertes Setup
        - einfachere Erweiterung
        - Zugang zu einzelnen library Komponenten

- Diagramme können importiert werden (Beispiel dafür unter erstem Link)
- Stylesheets inbegriffen
- umfangreiche Beispiele
- diagram-js wird für das eigentliche Zeichnen verwendet
    - Toolbox für interaktive Darstellung und Modifizierung von Diagrammen im Web
    - enthält Datenmodell für Beziehungen zwischen grafischen Elementen

D3-js

https://github.com/d3/d3
Demo Übersicht: https://github.com/d3/d3/wiki/Gallery

- Bibliothek wurde unter BSD Lizenz (https://opensource.org/licenses/BSD-3-Clause) veröffentlicht
- In reinem JavaScript programmiert
- Sehr einfache Einbindung per HTML Tag::
  ```
  <script src="https://d3js.org/d3.v5.min.js"></script>
  ```
- Library zum vereinfachten Ansprechen, Bearbeiten und Erstellen von HTML, SVG und CSS Elementen
- Diesen Elementen können Daten hinzugefügt werden, mit denen man Operationen (bspw. unsere Optimierungsfunktion) durchführen kann
- Beispielsweise lassen sich mit dem richtigen Datensatz über einfache Methoden SVG Graphen erstellen und bearbeiten
- Umfangreiche Dokumentation mit zahlreichen Beispielen
- Große Community (34tsd Fragen auf Stackoverflow)
- Viele YouTube Tutorials
- Viele Plugins verfügbar - Bsp: https://www.d3-graph-gallery.com/network.html
- Mit minimalem Overhead ist D3 extrem schnell und unterstützt große Datenmengen und dynamisches Verhalten für Interaktion und Animation. Der funktionale Stil von D3 ermöglicht die Wiederverwendung von Code durch eine vielfältige Sammlung von offiziellen und von der Community entwickelten Modulen
- Zur Darstellung werden keine neuen Standards verwendet, sondern die hauseigenen Mittel von HTML, CSS und SVG verwendet
- Styling kann per CSS einfach verändert werden
- Einfaches Debugging: Da die Knoten per browserbasierten Code dargestellt werden, kann der native Browser-Inspektor verwendet werden
- Transitionen & Animationen sind vereinfacht
- Bsp für einfachen Graphen: http://bl.ocks.org/benzguo/raw/4370043/
- Library für Export in PNG, SVG und PDF ist vorhanden: https://d3export.housegordon.org/
- Da Daten im SVG Format dargestellt werden, hätte man schon eine XML Export Struktur - Import lässt sich auch umsetzen (gibt bereits Anleitungen)
- Anlegen von Templates wäre via defs & use möglich: http://www.svgbasics.com/defs.html

JavaScript InfoVis Toolkit (jit)

http://philogb.github.io/jit/
Dokumentation: http://philogb.github.io/jit/static/v20/Docs/
Demos: http://philogb.github.io/jit/demos.html
http://philogb.github.io/jit/static/v20/Jit/Examples/ForceDirected/example2.html
Es ist unter Open Source(BSD) Lizenz und Teil der Sencha Labs foundation.
Die nötige Einarbeitungszeit kann ich nicht wirklich einschätzen, aber die Dokumentation ist übersichtlich und hat jede Menge Beispiele weshalb es auf mich den Eindruck macht, als wenn sich jemand mit JavaScript- Erfahrung recht schnell einarbeiten könnte.

Cytoscape.js

https://github.com/cytoscape/cytoscape.js
Doku: https://js.cytoscape.org/#introduction/factsheet
Demo: https://cytoscape.org/cytoscape.js-popper/

- vollumfassende Grafik library, diese erlaubt das anzeigen und manipulieren von interaktiven Graphen
- framework beinhaltet Analysetools aus der Graphentheorie (evt. nur mittels node.js)
- bietet einen breiten Umfang an Graphenarten (gerichtete, unterrichtete, Hypergraphen usw.)
- basiert rein auf javascript
- Open-Source (MIT-Lizenz)
- laut eigenen Angaben stark optimiert
- wird von allen modern Browsern unterstützt
- bietet canvas support
- gute umfassende Dokumentation die die Einarbeitung erleichtert bzw. die Arbeit mit Cytoscape (inklusive Codebeispielen und Beispielgraphen) und kein Thema auslässt
- rendern von Bildern der Graphen in verschiedenen Formaten möglich (nützlich für den export der Graphen als Bild)
- eingebaute Unterstützung von Gestensteuern (Desktop & Mobil)
- volle Kompatibilität (Im/Export) mit JSON
- regelmäßige Erweiterung bzw Wartung des Frameworks, also bestehender Support (regelmäßige commits auf github)
- relativ große Community (ca. 7k auf github)
- abseits der Doku gibt es viele Tutorials und Hilfen
- es gibt eine große Menge an Extensions
- laut eigene Angaben sinkt die Performance mit steigender Anzahl an Knoten (bzw. generellen Graphgröße)