ARTRAGIS

Zeste de savoir

Faire des diagrammes dans un tuto : que faire sur zds

8 novembre 2020

Table des matières

1. Mermaid JS: tout demander aux navigateurs 1
   1.1. Présentation 2
   1.2. Intégration à ZDS ? 2
   1.3. Yet another language 3
2. Une alternative: graphviz 3
3. On fait quoi ? 5

Ces derniers temps, je reprends un peu de motivations pour participer techniquement à ZDS. Et un sujet me plaît particulièrement: intégrer la possibilité de faire des diagrammes avec ZDS.

Des diagrammes il en existe des tonnes très connus:

— diagramme de flux (pour expliquer un algorithme ou même un processus comme un processus législatif par exemple)
— diagramme de classe (pour la programmation)
— diagramme de gantt (pour faire de la gestion de projet)
— …

? Problématique

Comment intégrer à zds la possibilité de créer un diagramme pour ensuite le rendre partout de la même manière tant sur le site web que dans nos PDF ?

Il existe probablement des milliers de manières de répondre à cette problématique, mais pour ce billet, je vais me concentrer sur une technique qui est particulièrement adaptée à zds: on écrit le diagramme avec une sorte de pseudo code plus ou moins simple dans le tuto lui-même et ZDS se charge de le transformer en une belle image.

Et vous allez voir que ça va être bien plus compliqué à réaliser qu'à exprimer :).

1. Mermaid JS : tout demander aux navigateurs

Heureusement pour nous, nous ne sommes pas les premiers à penser à ce genre de solutions. Dans le monde du JavaScript, a été développé Mermaid JS qui se présente comme "Une syntaxe à la Markdown pour faire des diagrammes".

1.1. Présentation

Ce logiciel, initié par Knut Sveidqvist vous permet de coder 7 ”types” de diagrammes:

— Des diagrammes de flux
— Des diagrammes de séquence
— Des diagrammes de classe
— Des diagrammes d'état
— Des diagrammes de Gantt
— Des diagrammes Entité/Relation
— Des graphiques en camembert

Vous entrez un texte dans une div qui a la classe `mermaid` et le script va vous traduire ça en image vectorielle: vous pouvez donc zoomer à l'infini sur un diagramme (et c'est parfois très utile).

Par exemple:

```
graph TD
  A[Christmas] --Get money--> B(Go shopping)
  B --> C{Let me think}
  C -->|One| D[Laptop]
  C -->|Two| E[iPhone]
  C -->|Three| F[fa:fa-car Car]
```

Listing 1 – Crée un diagramme qui en partant de l'idée que c'est Noël cous donne les marche à suivre pour vous acheter un cadeau.

Génère cette image vectorielle:

FIGURE 1.1. – Un diagramme de flux pour faire des achats de Noel

Le tout vient avec un système de thème qui permet de personnaliser l'affichage global des diagrammes.

1.2. Intégration à ZDS ?

Comment pourrions-nous intégrer mermaid à ZDS ?
La méthode la plus simple consiste à proposer aux gens d'écrire le code que j'ai proposé juste au dessus en utilisant la syntaxe markdown pour ajouter un code, ainsi nous aurions:

```
graph TD
  A[Christmas] --Get money--> B(Go shopping)
  B --> C{Let me think}
  C -->|One| D[Laptop]
  C -->|Two| E[iPhone]
  C -->|Three| F[fa:fa-car Car]
```

Ensuite, techniquement parlant, nous aurions à dire à zMarkdown que les code dont le langage est "mermaid" doivent juste être mis tels-quels dans un div. Puis du côté de zds-site, ajouter le CSS et le code de mermaid.
Cela nous pose deux soucis:

— L'interprétation du code pour le transformer en diagramme est potentiellement longue. Certains se souviennent peut être de l'époque où nos tutos de science passaient leur temps à faire des allers/retours sur le scroll à cause de l'interprétation des maths par Mathjax. On aurait le même soucis ici. Chaque graphique prend 1 à 10 secondes pour être transformé (j'utilise personnellement mermaid pour faire des graphes au boulot).
— Comment faire pour avoir l'image générée en latex et ensuite dans notre PDF ? (accessoirement le soucis serait identique dans le ebook) ?

En effet mermaid a un besoin absolu d'avoir un navigateur pour fonctionner.
La seule possibilité qui existe alors pour faire un rendu "côté serveur" est d'utiliser un navigateur *headless* pour que notre serveur puisse générer l'image.
Seulement ces outils sont souvent soit marqués comme outil de test (donc peu adaptés à la production) soit très gourmands en ressources et demandent d'avoir une interface graphique virtuelle telle que X-Virtual Frame Buffer(xvfb) pour fonctionner. En somme c'est techniquement possible mais il faut être très prudent avec ce qu'on fait.
Une autre solution serait de passer par un *langage intermédiaire* qui lui serait adapté au *rendu côté serveur* mais n'aurait pas besoin de navigateur. On y reviendra plus tard car d'autres considérations nous intéressent pour l'instant.

1.3. Yet another language

Venir rédiger sur ZDS c'est apprendre à utiliser markdown, et surtout le markdown sauce ZDS avec ses extensions pour les acronymes, les tableaux, les légendes, les blocs spéciaux…
Ajouter la possibilité de faire des diagrammes par texte, c'est demander d'apprendre un autre langage en plus du markdown. Cela pose quelques questions d'un point de vue utilisabilité:

— Comment présenter ça aux gens dans notre interface ? Comment faire une aide qui soit utilisable ? Comme beaucoup d'outils techniques, mermaid a une documentation en anglais. Seul son live editor permet d'ajouter un peu d'aide. En terme de réflexion, cela peut être intéressant, comme ce dernier est open source on peut imaginer que notre éditeur ouvre une "fenêtre d'édition" qui lance une instance du live editor puis, grâce aux nouveautés du JS modernes on pourrait proposer d'insérer le code pré-généré dans le tuto. Mais cela demande quand même aux futurs auteurs de coder avec un nouveau langage.
— Comment assurer l'accessibilité du diagramme ? Est-ce que nos légendes style `Code: ma légende` seront-elle suffisante pour rendre le diagramme généré accessible ?
— Autre chose ?

2. Une alternative : graphviz

Dans la liste des outils déjà existant on trouve aussi graphviz qui se présente comme un outil de positionnement et rendu des graphes.
Il est mille fois plus connu que mermaid, mais son expression est beaucou, beaucoup, plus difficile. Et pour cause: il s'agit d'organiser des graphes qui seront rendus par un moteur laissé libre (voir le tutoriel sur VisNetwork qui se base sur graphviz pour définir ses graphes). Le langage (nommé Dot) sur lequel il se base ne permet de définir que des graphes "simples" et

"dirigés". A vous de voir si votre graphe sera plutôt un diagramme de flux ou d'état une fois rendu.
Prenons par exemple un diagramme qui ressemble énormément au diagramme de flux de tout à l'heure:

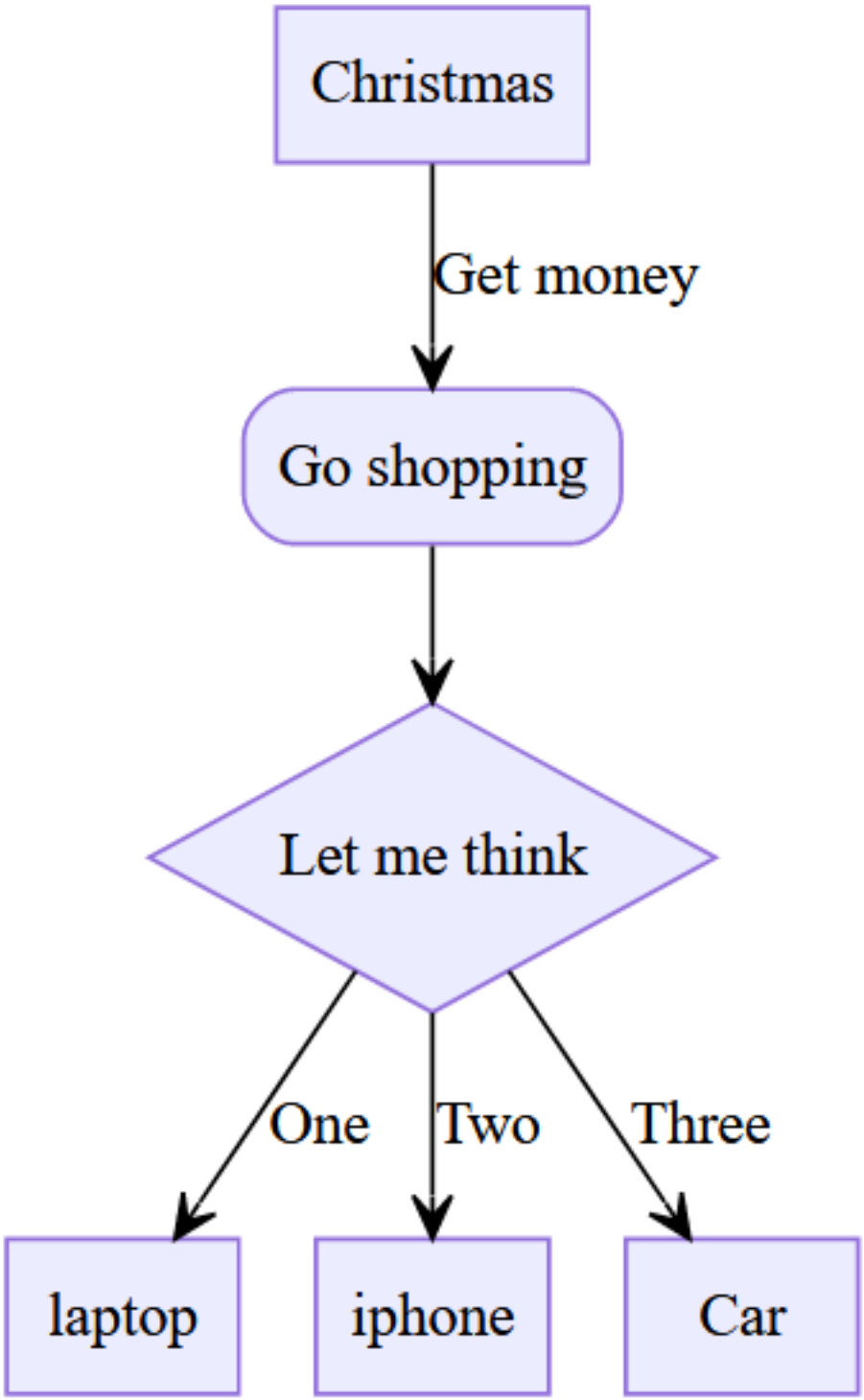

FIGURE 2.2. – Diagramme de flux avec GraphViz

Outre que le résultat est moins beau, pour le réaliser il aura fallu que je fasse un code beaucoup plus complexe:

```
digraph test {
    rankdir="TD"
    splines=polyline
    node[style=filled, fillcolor="#ECECFF", color="#9370DB"]
    edge[labelangle=90, labeldistance=2.5, arrowhead="open"]
    A[label=Christmas, shape=rect];
    B[shape=rectangle, label="Go shopping",
        style="rounded,filled"];
    C[shape=diamond, label="Let me think", height="1em"];
    D[shape=rect, label="laptop"];
    E[shape=rectangle, label="iphone"];
    F[shape=rect, label="Car"];
    A -> B [label="Get money", labelangle=20, labeldistance=2.5]
    B -> C
    C -> D[label="One"]
    C ->E[label="Two"]
    C ->F[label="Three", K=30]

}
```

Listing 2 – code pour générer le graphe du dessus

Bien sûr, ici je peux sélectionner mon ”moteur de rendu” et render-dot permet de faire un rendu en SVG (vectoriel) ou PNG côté serveur sans aucun besoin de navigateur *headless*.
Une hypothèse de travail pourrait donc être de transformer les diagrammes mermaid en dot pour ensuite les rendre.
Le principal problème c'est qu'a priori, tous les diagrammes supportés par mermaid ne sont pas des graphes donc il sera compliqué de les traduire en Dot. De plus comme vous l'avez remarqué, le rendu n'est pas aussi qualitatif.
Cependant graphviz possède quelques autres avantages:

- la granularité de la personnalisation est immense: une dizaine de formes de flèche contre 3 sur mermaid, des formes de liens droits, courbes, brisées alors que mermaid définit lui-même (parfois un peu bizarrement) s'il doit faire simplement droit ou brisé)
- la capacité à mettre deux annotations par flèche (au début et à la fin de la flèche, pour les cardinalités par exemple)
- La gestion des sous-graphes est aussi plus simple
- Il possède un éditeur simple en mode GUI: graphviz visual editor

3. On fait quoi ?

Je n'ai pas encore de réponse précise à cette question. Je ne sais pas vraiment qui va utiliser parmi les auteurs la capacité à faire des diagrammes. Et parmi ces diagrammes lesquels seront vraiment utiles.
De même je ne sais pas si finalement intégrer des liens vers ces outils ne serait pas plus simple.
Enfin, je ne sais pas non plus s'il n'y a pas quelqu'un qui a déjà codé quelque chose qui répondrait en tout point à notre besoin.
Je suis ouvert à toutes les suggestions mais aussi à toutes les réflexions. Si vous avez des idées d'intégration, de ce qui est important pour l'expérience utilisateur etc. N'hésitez pas à poster

un message dans les commentaires.