Plan de développement

Programmation de composants Angular pour la mise en œuvre de tests utilisateurs

Benjamin Mitton

Cédric Pinard

Université Grenoble Alpes

Sommaire

Introduction

Ce document décrit l'ensemble des fonctions utilisés et les algorithmes correspondant pour chacune d'entre elles. Ces fonctionnalités sont séparées en deux parties, la première partie et celle de l'affichage d'une linechart et la deuxième partie concerne toutes les fonctionnalités liées à la navigation au sein du composant. Chacune des fonctionnalités est définie par ses variables globales, ses entrées et sorties, son algorithme ainsi qu'une courte description de ce que réalise cette fonction.

Concepts de base

Ce découpage fonctionnel n'est possible que dans une architecture Angular, il faut donc connaître les liens entre les fichiers HTML et les fichiers TypeScript pour comprendre comment fonctionne certaines fonctions. Une partie des éléments utilise aussi le format de données Scalable Vector Graphics (SVG). Ce format utilisable avec le TypeScript permet d'afficher des ensembles graphiques vectoriels.

Conception détaillée

Afficher une linechart

Description

En tant que client, je souhaite que les données transmises au composant soit transformées en linechart, afin de pouvoir les analyser visuellement.

Objectifs

Afficher une linechart correspondante aux données transmises en paramètre.
La linechart doit être au plus possible paramétrable par l'individu.
La linechart doit permettre d'afficher une ou plusieurs lignes de données en son sein.
L'échelle des axes doit être paramétrable ou automatique.
La ligne de données doit pouvoir s'afficher de manière optimale pour des données discrètes ou continues.
Une configuration de base en cas de non remplissage des paramètres doit être présente.

Bibliothèques et librairies utilisées

- D3js pour créer le composant de linechart
- Les fonctions de cycle de vie d'Angular core pour gérer le dynamisme de la page.

Description des entrées et sorties

En entrée on retrouve les données et les paramètres du composant HTML.
Les données d'entrées sont sous le format suivant :

```
export interface Data {
  label: string;
  values: [number,number][];
  color: string;
  style: "line" | "area" | "both";
  interpolation: "linear" | "step";
}
```

Les paramètres du composant sont les suivants :
`@Input() width: number` = la largeur du composant qui contient la linechart
`@Input() height: number` = la hauteur du composant qui contient la linechart
`@Input() data: Data[]` = tableau de données sous le format expliqué ci-dessus
`@Input() domain: [number, number]` = la spécification de l'échelle de l'axe des ordonnées sur lequel seront représentées les données

Variables globales

`public title : string` = le titre du composant
`private margin : {top : number, right : number, bottom : number, left : number}` = les marges internes à la linechart
`private minTime: number` = la valeur minimale de temps récupérer sur les données
`private maxTime: number` = la valeur maximale de temps récupérer sur les données
`private svgHeight: number` = la hauteur du SVG sans les marges
`private svgWidth: number` = la largeur de la courbe sans les marges
`private scaleX: ScaleTime<number,number>` = définition de l'axe X
`private scaleY: ScaleLinear<number,number>` = définition de l'axe Y
`private svg: any` = la variable contenant la référence du SVG construit
`private area: d3.Area<[number, number]>[]` = tableau contenant les fonctions de création des areas
`private line: d3.Line<[number, number]>[]` = tableau contenant les fonctions de création des lines

Fonctions composites

ngAfterViewInit()

Variables globales présentes :

- svgHeight
- svgWidth
- area
- line

Description : Cette fonction proposée dans le cycle de vie d'Angular permet d'initialiser des variables après la création de la page et donc de récupérer les paramètres données dans l' HTML. On initie les variables globales.

Sortie : Aucune

Appel fonction :

- drawAxis()
- drawLineAndPath()
- buildStyleData()

Algorithme :

si timeline != vide
    alors on initie svgWidth et svgHeight par rapport aux paramètres du composant en retirant les marges
pour chaque jeu de données on appelle la fonction buildStyleData.

ngOnChanges(changes : SimpleChanges)

Variables globales présentes : Aucune

Description : C'est une fonction proposé par Angular qui s'active automatiquement en cas de modification de paramètres. Modifie la linechart en cas de changement sur data.

Sortie : Aucune

Appel de fonction : updateChart()

Algorithme :

si data change et que ce n'est pas la première fois
    alors on appelle updateChart()

buildEvent()

Variables globales présentes :

- timeline
- svg

Description : On initialise, la variable SVG avec l'élément SVG référencé dans le HTML au nom de timeline.

Sortie : Aucune

Appel de fonction : Aucun

Algorithme : svg = timeline

drawAxis()

Variables globales présentes :

- scaleX
- scaleY
- svg
- domain
- data

Description : Construit les axes X et Y en fonction de la variable domain si elle est spécifiée en paramètre sinon en fonction de la variable data et ajoute les axes créés dans le SVG.

Sortie : Aucune

Appel de fonction : Aucun

Algorithme :

pour scaleX on initialise le scaling
    on initialise le domain et la range
pour scaleY on initialise le scaling
    on initialise la range
si domain == undefined
    alors on initialise le domain par rapport aux données
sinon
    on initialise le domain grace a la variable domain
on ajoute les axes x et y au svg

drawLineAndPath()

Variables globales présentes :

- svg
- line
- area
- data

Description : Cette fonction créer les lignes et les aires correspondantes aux données et les ajoute dans le SVG.

Sortie : Aucune

Appel de fonction : Aucun

Algorithme :

Pour chaque tableau de données dans data
    si data[i] .style = line || both
        on ajoute une ligne au svg
    si data[i].style = area|| both
        on ajoute une area au svg

buildStyleData(element : Data, index : number)

Variables globales présentes :

- line
- area

Description : Ajoute une ligne à la variable globale line et/ou une aire à la variable globale area en fonction du style du paramètre element à l'index correspondant au paramètre index. L'interpolation de la courbe (aire ou ligne) est aussi précisée.

Sortie : Aucune

Appel de fonction : Aucun

Algorithme :

    si data[i].style = area || both
        si data[i].interpolation = step
            alors on créer l'area correspondante
        sinon

alors on créer l'area correspondante
si data[i].style = line || both
si data[i].interpolation = step
alors on créer la line correspondante
sinon
alors on créer la line correspondante

updateChart()

Variables globales présentes :

- data
- line
- area
- scaleX
- scaleY
- domain
- svg

Description : Cette fonction est appelée lorsque que le paramètre data change. On réinitialise les variables globales concernant les courbes et les axes afin de les adapter aux nouvelles données.

Sortie : Aucune

Appel de fonction :

- buildStyleData()
- updateLine()

Algorithme :

pour chaque jeu de données
on appelle buildStyleData()
si data[i].style = line
alors on remove les area du svg
si data[i].style = area
alors on remove les lines du svg
pour scaleX on initialise le scaling
on initialise le domain
on initialise la range
pour scaleY on initialise le scaling
on initialise la range
si domain == undefined

alors on initialise le domain par rapport aux données
sinon
on initialise le domain grâce à la variable domain
on ajoute les nouveaux axes au svg

updateLine()

Variables globales présentes :

- data
- line
- area
- svg

Description : Ajoute les nouvelles lines et/ou areas dans la variable svg.

Sortie : Aucune

Appel de fonction : Aucun

Algorithme :

pour chaque jeu de données
si data[i].style = area||both
modifie le svg avec la nouvelle area et ses paramètres
si data[i].style = line|| both
modifie le svg avec la nouvelle line et ses paramètres

Naviguer au sein de la linechart

Description

En tant que client, je souhaite pouvoir zoomer et me déplacer au sein des données affichées, afin de gagner en précision .

Objectifs

Zoomer selon l'axe des abscisses (temps).
La linechart possède une scrollbar horizontale pour me déplacer dans le temps.
On peut lier le zoom et la scrollbar de deux linecharts différentes.
Il existe une barre verticale déplaçable au clic qui représente le temps actuel.
Une bulle d'information sur les données en fonction de la position de la souris doit s'afficher.

Bibliothèques et librairies utilisées

- D3js pour créer le composant de linechart
- Les fonctions de cycle de vie d'Angular core pour gérer le dynamisme de la page.

Description des entrées et sorties

En entrée on retrouve ces données passée en paramètre du composant :
`@Input() range: [number,number]` = plage données à afficher
`@Input() currentTime: number` = temps actuel
`@Input() speedZoom: number` = vitesse de zoom
En sortie on retrouve :

```
@Output() rangeChange = new EventEmitter<[number,number]>();
@Output() currentTimeChange = new EventEmitter<number>();
```

Variables globales

`private dataZoom: Data[]` = vue locale des données modifiées avec le zoom et/ou la scrollbar
`private idZoom: number` = contient le nombre de cran de molette utilisé
`private tooltip!: Selection<SVGGElement,unknown,null,undefined>` = contient la référence au tooltip
`private lastDatalength:number` = dernière taille de plage de temps avant modification
`private modeToolTips: "normal" | "inverse"` = représente les deux modes d'affichage du tooltip
`private lengthTime: number` = longueur en ms de la plage de données affichées
`private currentTimeSelected: boolean` = vrai si le clic est effectué sur la currentTime
`private scrollbarSelected: boolean` = vrai si le clic est effectué sur la scrollBar
`private lastPos: number` = dernière position de la souris par rapport a la scrollBar

Fonctions composites :

ngOnInit()

Variables globales présentes :

- dataZoom
- lastDatalength
- data
- title
- data

Description : Fonction du cycle de vie d'Angular s'exécute à l'initiation du contenu de la page. On l'utilise pour instancier les variables globales par rapport aux paramètres transmis par l'utilisateur.

Sortie : Aucune

Appel de fonction : Aucun

Algorithme :

dataZoom = copie de data
lastDatalength = taille de dataZoom
pour chaque jets de données dans data
    si i == taille de data-1
        title = title + data[i].label+ '.'
    sinon
        title = title + data[i].label + ','

ngAfterViewInit()

Variables globales présentes : Aucune

Description : Complète la première partie détaillée dans le doc 1. On appelle les fonctions de création des éléments de navigation.

Sortie : Aucune

Appel de fonction :

- buildZoom
- buildEvent
- drawTooltips

- drawLineCurrentTime
- buildScrollBar

Algorithme :

buildZoom()
buildEvent()
drawTooltips()
drawLineCurrentTime()
buildScrollBar()

ngOnChanges(changes : SimpleChanges)

Variables globales présentes :

- idZoom
- range
- dataZoom
- lengthTime

Description : Complète la première partie détaillée dans le doc 1. Fonction du cycle de vie prend en compte les changements de données transmises en paramètre.

Sortie : Aucune

Appel de fonction :

- updateDataZoom
- updateCurrentTime
- updateSvg
- controlRange

Algorithme :

si (il y a un changement et que ce n'est pas le premier et que range[0]!=0 et que range[1]!= 0)
ou (la range a changé et que ce n'est pas la première fois)
    on calcule idZoom
    range = controlRange(range[0], range[1]-range[0])
    si taille de data!= 0
        updateDataZoom(range[0],range[1])
        updateSvg(range[0],range[1])
si il y a changement currentTime et ce n'est pas la premiere fois et que la taille de data!= 0
    alors updateCurrentTime()

buildEvent()

Variables globales présentes :

- svg
- currentTimeSelected

Description : Complète la première partie détaillée dans le doc 1. On ajoute les événements en cas de déplacement, de clic ou d'action de la molette, de la souris.

Sortie : Aucune

Appel de fonction :

- moveCurrentTime
- showInfo
- activeZoom
- hideInfo

Algorithme :

```
en cas de déplacement de la souris a l'intérieur du svg
        alors si la barre verticale est sélectionné
                alors movecurrentTime()
        sinon showInfo()
en cas de sortie de la souris du svg
        alors currentTimeSelected = false
                hideInfo()
en cas de mouvement de la molette
        activeZoom()
en cas de lever du clic de la souris
        currentTimeSelected = false
```

drawLineCurrentTime()

Variables globales présentes :

- data
- currentTime
- svg
- svgHeight
- currentTimeSelected

Description : Créer et ajoute au SVG la barre verticale représentant le temps actuel.

Sortie : Aucune

Appel de fonction : hideInfo()

Algorithme :

si data non vide
    alors si currentTime = 0
        alors curentTime est égale au timestamp le plus faible de data
on ajoute au svg la ligne verticale
on ajoute au svg le cercle
au clic sur le cercle
    currentTimeSelected = true
    hideInfo()

drawScrollBar()

Variables globales présentes :

- zoneScrollbar
- scrollbar
- renderer

Description : Dessine la scrollbar horizontale et ajoute les événements en cas de clic et de mouvement de la souris.

Sortie : Aucune

Appel de fonction :

- activeScrollbar
- desactiveScrollbar
- updateRange

Algorithme :

si on clic sur la scrollbar alors
    activeScrollbar(event)
si on la souris quitte le composant ou si l'on relâche le clic alors
    desactiveScrollbar
si l'on déplace la souris dans le composant alors
    updateRange(event)

updateChart()

Variables globales présentes :

- dataZoom
- svg
- lastDatalength
- minTime
- maxTime

Description : Complète la première partie détaillée dans le doc 1. On met à jour les variables globales nécessaires à la navigation.

Sortie : Aucune

Appel de fonction :

- drawLineCurrentTime
- updateScrollbar
- updateTooltips
- buildZoom

Algorithme :

dataZoom = copie de data
buildZoom()
supprime la ligne verticale représentant le temps actuel
drawLineCurrentTime()
updateScrollbar(minTime,maxTime)
updateToolTips()
lastDatalength=dataZoom.length

updateLine()

Variables globales présentes : dataZoom

Description : Complète la première partie détaillée dans le doc 1. On recréer les lignes et/ou les aires en fonction des nouvelles données mais aussi du zoom actuel.

Sortie : Aucune

Appel de fonction : Aucun

Algorithme : le même que le précédent mais cette fois on remplace data par dataZoom

updateSvg(min : number, max : number)

Variables globales présentes :

- scaleX
- svg

Description : Actualise tout le SVG (axe des abscisses, les lignes et/ou les aires, la barre verticale représentant le temps actuel et la scrollbar horizontale)

Sortie : Aucune

Appel de fonction :

- updateLine
- updateCurrentTime
- updateScrollbar

Algorithme :

on adapte l'echelle de l'axe X en fonction des paramètres min et max puis on l'ajoute au svg
updateLine()
updateCurrentTime()
updateScrollbar(min,max)

updateCurrentTime()

Variables globales présentes :

- currentTime
- svg

- dataZoom
- scaleX
- scaleY

Description : Cette fonction sert a redessiner la barre verticale currentTime lors d'un déplacement de celle-ci ou de la plage de temps afficher.

Sortie : Aucune

Appel de fonction : Aucun

Algorithme :

on modifie la barre verticale en fonction de currentTime, scaleX, scaleY et on l'ajoute au svg
si le temps actuel est compris dans la plage de temps afficher (dataZoom)
    alors on affiche la barre
    alors on affiche le cercle de la barre
sinon
    alors on n'affiche pas la barre
    alors on n'affiche pas le cercle de la barre

updateScrollbar(min : number, max : number)

Variables globales présentes :

- scrollbar
- svgWidth
- minTime
- lenghTime

Description : On redéfinit la taille et la position de la scrollbar horizontale.

Sortie : Aucune

Appel de fonction : Aucun

Algorithme :

scrollbar.marginLeft = svgWidth*(min-minTime)/lengthTime
scrollBar.width = svgWidth*(max-min)/lengthTime

activeScrollbar(event : MouseEvent)

Variables globales présentes :

- scrollbarSelected
- lastPos
- margin.left

Description : On change la valeur du booléen scrollbarSelected à vrai et on enregistre la position actuelle de la souris.

Sortie : Aucune

Appel de fonction : Aucun

Algorithme :

scrollBarSelected = true
lastPos = event.posX-margin.left

desactiveScrollbar()

Variables globales présentes :

-scrollbar
-lastPos

Description : On change la valeur du booléen scrollbarSelected à faux et lastPos à 0.

Sortie : Aucune

Appel de fonction : Aucun

Algorithme :

scrollbarSelected = false
lastPos = 0

updateRange(event : MouseEvent)

Variables globales présentes :

- scrollbarSelected
- range
- rangeChange

- dataZoom
-margin.left

Description : Cette fonction est appelée lorsque l'on déplace la scrollbar horizontale. Elle calcule le déplacement de la plage de temps affichée en fonction du mouvement de la souris et émet la nouvelle plage de temps via le output rangeChange.

Sortie : Aucune

Appel de fonction :

- controlRange
- updateDataZoom
- updateSvg

Algorithme :

si scrollbarSelected
    on supprime les effets de l'event non désiré
    on calcul la plage de données à afficher en fonction du mouvement de la souris
    on obtient donc minLocalTime et lengthLocalTime
    on contrôle la range : range = controlRange(minLocalTime,lengthLocalTime)
    updateDataZoom(range[0],range[1])
    updateSvg(range[0],range[1])
    on émet la nouvelle range via rangeChange

buildZoom()

Variables globales présentes :

- minTime
- maxTime
- lengthTime
- idZoom
- data

Description : On initialise des variables globales permettant la navigation.

Sortie : Aucune

Appel de fonction : Aucun

Algorithme :

on initialise minTime, maxTime et lengthTime avec data

on intialise idZoom à 0

updateToolTips()

Variables globales présentes : tooltip

Description : On détruit et reconstruit la bulle d'information.

Sortie : Aucune

Appel de fonction : addToolTips

Algorithme :

détruire le tooltips
drawToolTips()

drawToolTips()

Variables globales présentes :

- svg
- modeToolTips
- dataZoom

Description : On créer une bulle d'information qui s'affiche différemment en fonction des données sur la courbe à l'endroit ou à l'envers.

Sortie : Aucune

Appel de fonction : Aucun

Algorithme :

on ajoute le tooltip au svg
on ajoute un petit cercle au tooltip
on ajoute un deuxieme cercle au tooltip
si modeToolTips = normal
    alors on ajoute le contour
    pour chaque jeu de données
        on ajoute le texte correspondant
sinon
    alors on ajoute le contour

pour chaque jeu de données
    on ajoute le texte correspondant

showInfo(event : MouseEvent)

Variables globales présentes :

- dataZoom
- tooltip
- modeToolTips

Description : Cette fonction est appelée quand la souris survole le SVG correspondant, elle a pour but de faire afficher la bulle d'information et de placer dedans les bonnes informations récupérées sur les données.

Sortie : Aucune

Appel de fonction : updateToolTips

Algorithme :

si dataZoom n'est pas vide
    alors pour chaque données de la première colonne
        on met cette donnée dans un tableau simple
    on rend le tooltip visible
    x0 = la position de la souris en valeur de temps
    x = l'indice du tableau ou se trouve la valeur la plus proche de x0
    si x > datazoom.length-1
        alors x = datazoom.length-1
    si x<0
        alors x=0
    d = la valeur a l'indice x
    t = le timestamp a l'indice x
    si le tooltip s'affiche trop haut
        si il n'est pas déjà inversé
            alors on change son mode en inverse
            updateToolTips()
    sinon
        si il n'est pas déjà normal
        alors on change son mode en normal
        updateToolTips()
pour chaque jeu de données dans dataZoom
    on récupère les valeurs dans le tableaux

on les assigne au balise texte du tooltip
on déplace le tooltip en fonction des valeurs de la souris

hideInfo()

Variables globales présentes : tooltip

Description : Cette fonction s'active quand la souris sort du SVG correspondant. Elle a pour but de supprimer l'affichage de la bulle d'information.

Sortie : Aucune

Appel de fonction : Aucun

Algorithme : changer le style du tooltip

activeZoom(event : WheelEvent)

Variables globales présentes :

- idZoom
- lengthTime
- dataZoom
- speedZoom
- range

Description : Cette fonction s'active quand l'utilisateur active la molette de la souris (peu importe son sens). Elle permet d'agrandir ou de réduire la plage de temps affichée.

Sortie : Aucune

Appel de fonction :

- controlRange
- updateDataZoom
- updateSvg

Algorithme :

On détermine idZoom (nombre de cran de molette)
On détermine la position de la souris
On détermine lengthLocalTime avec idZoom et speedZoom
Si la taille de la plage de temps est supérieure à 200 ms

alors on détermine minLocalTime grâce à la taille et la position de la souris
alors range = controlRange(minLocalTime, lengthLocalTime)
alors updatDataZoom(range[0],range[1])
alors updateSvg(range[0],range[1])
alors on émet range via l'output rangeChange
sinon
alors on reprend l'idZoom précédent

updateDataZoom(min : number, max : number)

Variables globales présentes :

- data
- dataZoom

Description : Cette fonction est appelée lorsqu'il y a un changement de données ou de plage de données. Elle a pour but de modifier dataZoom en fonction de la plage de données passée en paramètre.

Sortie : Aucune

Appel de fonction : Aucun

Algorithme :

Pour chaque jeu de données :
On copie les valeurs de data dans dataZoom si celles-ci sont supérieures à min et inférieures à max.
On ajoute à dataZoom une valeur dont le timestamp est égale à min et une égale à max en déduisant leur valeur de data .

moveCurrentTime(event : MouseEvent)

Variables globales présentes :

-margin
-dataZoom
-currentTime
-currentTimeChange

Description : Cette fonction est appelée lorsque l'on drag la barre verticale représentant le temps actuel. Elle permet de modifier la position d'affichage de cette barre, de modifier la variable globale currentTime et émet le nouveau temps actuel via l'output currentTimeChange.

Sortie : Aucune

Appel de fonction : updateCurrentTime

Algorithme :

On définit le temps actuel grâce à la position de la souris.
On vérifie que le temps actuel est bien compris dans la plage données.
updateCurrentTime() ;
On émet currentTimeChange.

controlRange(min : number, length : number)

Variables globales présentes :

- minTime
- maxTime

Description : Cette fonction contrôle que les paramètres sont corrects et elle retourne un min et un max ajustés si besoin.

Sortie : [min,max]

Appel de fonction : Aucun

Algorithme :

Si min<minTime
    alors min=minTime
max = min + length
Si max>maxTime
    alors max=maxTime
    alors min=max-length
Si min<minTime
    alors min=minTime
retourne [min,max]