Linaïa
Innover Tester Développer

Documentation Technique

CNRS

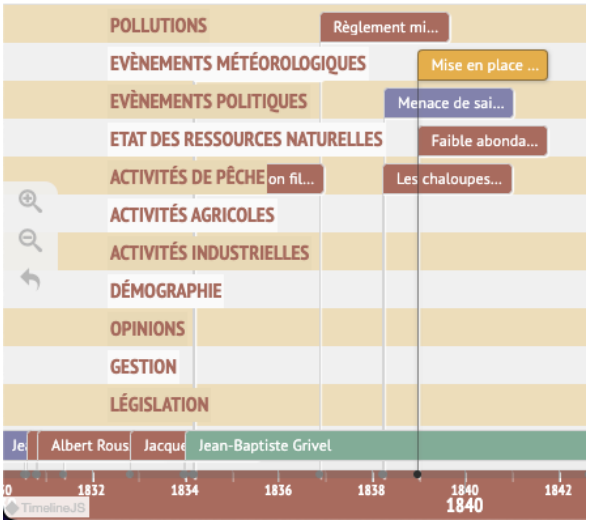
Amélioration TimeLineJs

Sommaire

Mise en avant des catégories

Initialisation du style

Affichage des noms de catégories au premier plan.



Ajout d'un padding sur chacune des catégories.

Attribution de la couleur de la ligne de catégorie à la couleur d'arrière-plan du texte.

Assombrissement de la couleur d'arrière-plan du texte pour être en adéquation avec la couleur de la ligne de catégorie.

Augmentation du z index TL-TimeGroup.less ligne 27

Code modifié :

Fichier TimeGroup.js

```
87    _initLayout () {
88
89        // Create Layout
90        this._el.message = DOM.create("div", "tl-timegroup-message ", this._el.container);
91
92        //Handle Color categories
93        this._el.message.style.backgroundColor = this.data.color;
94        this._el.message.style.filter = "brightness(98%)";
95        this._el.message.style.padding = "2px 2px 3px 2px";
96
97        this._el.message.innerHTML = this.data.label;
98    }
```

Gérer les catégories et leurs couleurs

Attribution de l'ordre et couleur des catégories

L'ordre et la couleur des catégories sont à définir dans le fichier Google Sheets, au tout début du tableau à partir de la ligne n°1.

Group	GroupColor	GroupOrder
Législation	#EFE1C2	11
Gestion		10
Opinions	#EFE1C2	9
Etat des ressources naturelles		4
Activités de pêche	#EFE1C2	5
Activités agricoles		6
Activités industrielles	#EFE1C2	7
Démographie		8
Pollutions	#EFE1C2	1
Evènements météorologiques		2
Evènements politiques	#EFE1C2	3

Récupération des valeurs

Fichier TimeScale.js

```
if (slides[i].group) {
    if (groups.indexOf(slides[i].group) < 0) {
        var colorLigne = slides[i].GroupColor;
        var group_order = slides[i].GroupOrder;
        colors.push(colorLigne);
        group_orders.push(group_order);
        groups.push(slides[i].group);
    }
```

Une fois les valeurs définies dans la table Google Sheets, elles sont récupérées et attribuées dans 2 tableaux colors et group_orders.

Ainsi pour chaque groupe existant, la couleur et l'ordre sont ajoutés.

group_info est accessible dans le fichier TimeNav.js ce qui permet de modifier la Timeline.

```
for (var i = 0; i < groups.length; i++) {
    group_info[i] = {
        label: groups[i],
        color: colors[i],
        order: group_orders[i],
        idx: i,
        positions: [],
        n_rows: 1, // default
        n_overlaps: 0
    };
}
```

Création des groupes

Fichier TimeNav.js

```
_createGroups() {
    this._groups = [];
    var group_labels = this.timescale.getGroupLabels();

    var group_ordered = this._reorderGroup(group_labels);
    if (group_labels) {
        this.options.has_groups = true;
        for (var i = 0; i < group_ordered.length; i++) {
            this._createGroup(group_ordered[i]);
        }
    }

}
```

Les valeurs des groupes sont récupérées dans le fichier TimeScale.js.

group_labels détient maintenant les groupes avec leur ordre et leur couleur.

Les groupes ne sont pas encore ordonnés. Création de group_ordered en appelant _redorderGroup.

La fonction _reorderGroup prend en paramètre les groupes et les réordonnent.

Une fois réorganisés, les groupes sont créés dans le fichier TimeGroup.js.

```
_reorderGroup(groups) {
    var group_ordered = [];
    //sort groups by group_order
    for (var i = 0; i < groups.length; i++) {
        var group = groups[i];
        if (group.group_order) {
            group_ordered[group.group_order - 1] = group;
        }
    }
    return group_ordered;
}
```

Fichier TimeMarker.js

```
_initLayout() {

    // Create Layout
    let isHidden = this.data.GroupOrder ? 'hidden' : '';
    this._el.container = DOM.create("div", `tl-timemarker ${isHidden}`);
```

Les groupes créés en début de table sont considérés par défaut comme des événements et de ce fait s'affichent avec un « marker » sous forme d'une boîte.

Ces boîtes doivent donc être cachées pour une meilleure lisibilité de la Timeline. Alors, dans la table Google Sheets, si un événement contient une valeur dans la colonne GroupOrder, cela masque automatiquement le marker dans la Timeline.

Attribution de la couleur

Fichier TimeGroup.js

```
// Data
this.data = {
    label: "",
    rows: 1,
    color: ""
};
```

On ajoute l'attribut couleur à la classe TimeGroup.

Puis, on y accède via this.data.color.

```
// Change the group line color
this._el.container.style.backgroundColor = this.data.color;
```

Gérer la couleur des markers

Initialisation

La couleur du marker est définie au format hexadécimal #RRVVBB dans la table Google Sheets, dans la colonne `MarkerColor`.

MarkerColor
#8db5a1
#b18869
#b16972
#b16972
#b18869
#b18869
#8db5a1
#8db5a1
#8db5a1
#8db5a1

Fichier ConfigFactory.js

```
function extractEventFromCSVObject(orig_row) {

    let row = {}
    Object.keys(orig_row).forEach(k => {
        row[k] = trim(orig_row[k]) // get rid
    })
    var d = {
        id: row['n°ID'] || '',
        type: row['Type'] || '',
        categories: row['Categorie'] || '',
        markerColor: row['MarkerColor'] || '',
        GroupOrder: row['GroupOrder'] || '',
        GroupColor: row['GroupColor'] || '',
        parentOf: row['parentOf'] || '',
        childOf: row['childOf'] || '',
        typeOfLink: row['TypeOfLink'] || '',
        ZoomOnClick: row['ZoomOnClick'] || '',
```

Une fois la couleur définie, on ajoute une propriété dans l'objet marker.

Cette propriété prend la valeur de row['MarkerColor'] cela correspond à la valeur inscrite dans la colonne `MarkerColor` dans la table Google Sheets.

S'il n'y a pas de valeur de couleur spécifiée, la case sera par défaut affichée en gris.

Fichier TimeMarker.js

Lors de l'initialisation du layout et des markers, leur couleur de fond est changée par la valeur inscrite dans la colonne `MarkerColor`.

```
// Handle color
this.data.markerColor != "" ? this._el.content_container.style.backgroundColor = this.data.markerColor : "";
```

Mise en avant des événements et de leurs enchainements

Relation entre Parent et Enfant

Pour pouvoir mettre en avant les événements et leurs enchainements, la relation entre les éléments s'active en listant les identifiants (id) dans les colonnes `parentOf` et `childOf`.

parentOf	childOf
	E0035
E0034,E0036	
	E0035

Un événement peut avoir plusieurs relations parent ou enfant.
Ainsi toutes les relations peuvent être mentionnées, il suffit de séparer les identifiants d'événements par une virgule ou un point-virgule dans les colonnes dédiées.

Ainsi au clic sur un événement dans la Timeline, tous les autres événements seront estompés pour faire ressortir uniquement les événements qui lui sont liés et indiqués dans les colonnes `parentOf` et `childOf`.

Récupération des informations

`Fichier` ConfigFactory.js

```
35          parentOf: row['parentOf'] || '',
36          childOf: row['childOf'] || '',
37      }
38      if(d.parentOf != ''){
39          var parents = d.parentOf.split(/[,;]/);
40          d.parentOf = parents;
41      }
42      if(d.childOf != ''){
43          var children = d.childOf.split(/[,;]/);
44          d.childOf = children;
45      }
```

Les champs sont récupérés dans la table Google Sheets puis affectés à parentOf et childOf.
Une fois récupérés, ils sont traités pour séparer les valeurs, via une virgule ou un point-virgule.

Interaction au clic

Fichier TimeMarker.js

```
_onMarkerClick(e) {
    //Remove highligh from other markers
    $(".tl-timemarker-content-container").removeClass('highlighted');
    $(this._el.content_container).addClass('highlighted');
```

Une classe highlighted est ajoutée au marker cliqué pour le faire ressortir.

Ensuite, mise en surbrillance (_setHighlight) de tous les markers liés grâce aux identifiants contenus dans les parentOf et childOf.

```
//highlight the parent and its children
this._setHighlight(this.data.parentOf);
this._setHighlight(this.data.childOf);
```

La fonction prend en paramètre data, et boucle sur data.
Pour chaque parent et/ou enfant, les espaces sont supprimés et la classe highlighted est ajoutée.

```
_setHighlight(data) {
    for (var i = 0; i < data.length; i++) {
        //remove space
        var str = data[i].replace(/\s/g, '');
        $("." + str).addClass("highlighted");
    }
}
```

"."+str Fait référence au marker, en effet lors de la création du marker, la valeur d'id contenue dans la table Google Sheets est attribuée à une id Marker

```
let idMarker = this.data.id;
idMarker = idMarker.replace(/\s/g, '');
this._el.content_container = DOM.create("div", `tl-timemarker-content-container ${idMarker} ${hasParent} ${isHidden} `, this._el.container);
```

Gestion de l'opacité des markers

Au clic, l'opacité change seulement si le marker contient des enfants ou des parents.

```
//Change opacity if marker has childs or parents
if (!this.data.parentOf.includes('') || !this.data.childOf.includes('')) {
    $(".tl-timemarker-content-container").not('.highlighted').not('.tl-timemarker-active').css('opacity', 0.1);
}
```

Relation distante

Lorsqu'un marker enfant est cliqué, les autres markers enfants de son parent sont affichés grâce à la méthode fire et au paramètre markerclick qui nous renvoie sur la méthode de TimeNav.js

Fichier TimeMarker.js

```
//Fire event that this marker has been selected
this.fire("markerclick", { unique_id: this.data.unique_id, zoomLevel: this.data.ZoomOnClick ,parent: this.data.childOf });
```

Une fois dans le fichier TimeNav.js _onMarkerClick est appelé.

Fichier TimeNav.js

```
marker.on('markerclick', this._onMarkerClick, this);
```

```
_onMarkerClick(e) {
    if (e.parent) {
        let parents = this._removeBlankSpace(e.parent);

        // Find the child of the parent and highligh it
        for (var i = 0; i < parents.length; i++) {
            let index = this._markers.findIndex(x => x.data.id == parents[i]);
            if(index == -1){
                //skip loop
                continue;
            }
            let childs = this._markers[index].data.parentOf;
            if (childs.length > 0) {
                childs.map((child) => {
                    //clear space
                    var str = child.replace(/\s/g, '');
                    $('.' + str).addClass('highlighted');
                })
            }
        }
    }
```

Cette méthode recherche pour chaque parent, les enfants de celui-ci.

Ensuite, il applique pour chaque enfant une classe highlighted.

La ToolTip

Si un marker est cliqué, une tooltip apparaît au survol (hover) de ses enfants.
Le texte de la tooltip est à définir dans la table Google Sheets dans la colonne TypeOfLink.

n°ID	parentOf	childOf	TypeOfLink
E0034		E0035	Conséquence
E0035	E0034,E0043		Cause indirecte
E0036		E0035	Cause direct

Dans cet exemple :
Engouement pour.... est E0035
La pêche aux coquilles... est E0034
Décret taill... est E0036

Fichier TimeMarker.js

Pour pouvoir créer une tooltip, la fonction _initiateToolTip est utilisée avec les parents et enfants concernés.

```
//initialization of tooltip on only parent and child
this._initiateTooltip(this.data.parentOf);
this._initiateTooltip(this.data.childOf);
```

```
_initiateTooltip(data) {
    if (!Array.isArray(data)) {
        return;
    }
    data.forEach(element => {
        var str = element.replace(/\s/g, '');
        $("." + str).mouseenter(function () {
            $(".tooltip-" + str).removeClass('tl-tooltip-hidden').addClass('tl-tooltip');
        })
        $("." + str).mouseleave(function () {
            setTimeout(function () {
                $(".tooltip-" + str).removeClass('tl-tooltip').addClass('tl-tooltip-hidden');
            }, 1000);
        }
        )
    });
}
```

La fonction change la classe de la tooltip des enfants et parents si la souris passe dessus.

De même si la souris en sort.

Le changement se fait via les classes tl-tooltip et tl-tooltip-hidden

Mise en place de la tooltip sur chaque élément possédant un TypeOfLink

```
if (this.data.typeOfLink != "") {

    this._el.tooltip = DOM.create("div", `tl-tooltip-hidden tooltip-${this.data.id}`, this._el.timespan);
    this._el.tooltip.innerHTML = this.data.typeOfLink;
}
```

On Blur

Lorsqu'un utilisateur clique sur un autre marker ou ailleurs sur la Timeline, les markers, les tooltips, et l'opacité sont réinitialisés.

Ces actions sont réalisées grâce à la fonction _onMarkerBlur voir ci-dessous.

Fichier TimeMarker.js

```
_onMarkerBlur(e) {
    this.fire("markerblur", { unique_id: this.data.unique_id });
    $(".tl-timemarker-content-container").css('opacity', 1);
    $(".tl-timemarker-content-container").removeClass('highlighted');
    this._deinitiateTooltip(this.data.parentOf);
    this._deinitiateTooltip(this.data.childOf);
}
```

```
_deinitiateTooltip(data) {
    if (!Array.isArray(data)) {
        return;
    }
    data.forEach(element => {
        data.forEach(element => {
            var str = element.replace(/\s/g, '');
            $("." + str).unbind('mouseenter');
        })
    });
}
```

La fonction _deinitiateTooltip permet d'enlever les binds de mousenter et mouseleave des tooltips devenu inutile.

Sans cela les tooltips resteraient affichés en survol même après avoir choisi un autre marker.

Mise en place d'un zoom lors d'un clic

Mise en place

Pour pouvoir afficher un zoom spécifique lors d'un clic sur un marker, dans la table Google Sheets, la colonne `ZoomOnClick` est renseignée et définit le niveau de zoom associé à l'événement.

ZoomOnClick
10
2
4
10

Par défaut, les intervalles de valeur de zoom sont compris entre 0 et 10, 10 représentant le niveau de zoom le plus élevé.

Récupération

Fichier ConfigFactory.js

Définition de ZoomOnClick pour la classe TimeMarker

```
ZoomOnClick: row['ZoomOnClick'] || '',
```

Fichier TimeMarker.js

Comme pour la relation distante, utilisation de la méthode fire avec this.data.ZoomOnClick en paramètre.

```
//Fire event that this marker has been selected
this.fire("markerclick", { unique_id: this.data.unique_id, zoomLevel: this.data.ZoomOnClick ,parent: this.data.childOf });
```

Puis l'on rajoute dans _onMarkerClick le niveau de zoom voulu.

```
// Set the zoom level
if (e.zoomLevel != "") {
    this.setZoom(e.zoomLevel);
}
```