05B - Travail 4 | JavaScript avancé

Consignes – Version A23.1.0

Mise en situation

Vous devez créer une application de style **Gestion de projet** (un peu comme le rôle d'un Trello ou d'un planner). Pour ce faire, vous devrez utiliser une **base de données** (structure d'objets et tableaux imbriqués) comme dans le TP3 qui contiendrait **5 tâches de départ**. Les tâches doivent apparaître sous forme de **cards** et une **bibliothèque JavaScript** devra être utilisée pour présenter les tâches sous forme de **diagramme de Gantt**.

Objectifs

a) Intégrer et utiliser une bibliothèque logicielle JavaScript
b) Utiliser des attributs HTML personnalisés
c) Utiliser les fonctions de minuterie
d) Ajouter de l'animation dynamiquement

Modalités de remise

Détails

- À faire en équipe de 2
- Toute forme de **plagiat** entraîne directement la note de zéro (0%) pour la remise;

Date et pondération

- Pondération sur la session : **15%**
- Remise avant le **10 décembre avant minuit**

Grille d'évaluation

- Voir le barème dans le dépôt sur Teams.

Fichier(s) et espace de dépôt GitLab

- ATTENTION à votre nom de projet **TP4-NOM1-Prenom1_NOM2- Prenom2**
- Remise par GitLab (voir en annexe)

TEAMS

- Déposer AUSSI une copie de votre site dans l'espace Devoir sur Teams (plan B).
- Dans le sous-dossier **« docs »**, assurez-vous de mettre un lien vers le Trello ou le Planner Vous devez reproduire et séparer la répartition des tâches (voir

Répartition des tâches à la page 7)

- **Aucun fichier CSS** : il sera généré à partir de votre fichier SASS tel que vu en classe.
- N'oubliez pas de **CONFIRMER VOTRE REMISE** Teams en appuyant sur le bouton Remettre

Réalisation du site

Versionnage avec Git

2.1.1 - Création du dépôt distant sur GitLab

Vous devez utiliser l'outil de versionnage **Git** dès le début du travail. Suivez les notes de cours pour pouvoir configurer Git afin de collaborer avec votre collègue (à faire par le membre responsable de l'équipe [Membre 1]).

1. ATTENTION au nom du projet (voir première page)
2. Créer le fichier « .gitignore »
3. Ajouter votre collègue avec le rôle « **Maintainer** »
4. Ajouter l'utilisateur « Morphiis » comme **« reporter »** dans la liste de membres.

2.1.2 - Suivi des modifications

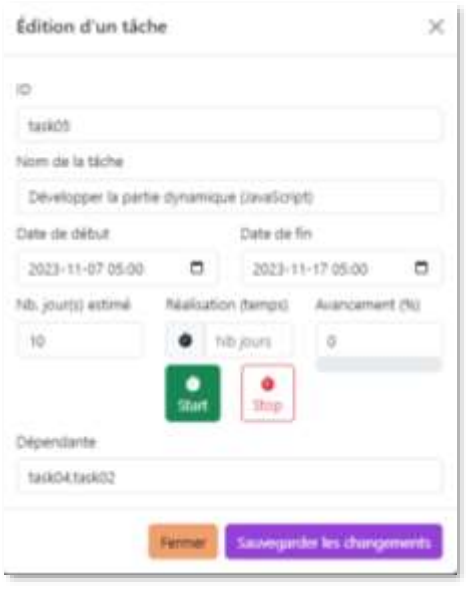
Tout au long de réalisation du site, vous devez envoyer votre code vers le dépôt distant. Voici les règles à suivre à ce sujet :

- Il doit y avoir au moins **1 commit** effectué par **séance de travail** (journée où vous travaillez sur le TP). De plus, il devrait y avoir au moins une séance de travail par semaine. La **rigueur*** dans le travail sera ainsi évaluée dans le cadre du TP.
- Il doit y avoir au moins **1 commit** pour chaque page du site
- Les **messages de commit** doivent être **significatifs** et contenir le nom de la page ou de la composante développée. Exemples de message :
  - Commande : Ajout des champs de formulaire
  - Accueil : Intégration de la mosaïque d'images
  - Header : Complétion de la navigation
- **Important** : N'oubliez pas d'envoyer vos commits vers le projet sur GitLab (**Push**) et de valider régulièrement le bon fonctionnement.

Note : La **rigueur dans la réalisation des tâches et le respect des règles et standards correspond à la compétence de **savoir-être** à évaluer dans le cadre du cours (tel que spécifié dans le plan de cours).*

Contraintes générales

- Créer votre **palette de couleurs** (http://colormind.io/bootstrap/) et réaffecter les $variables utiles (ex. : primary, secondary, etc.)
- Choisir des **polices** qui se marient bien et changer les variables correspondantes dans le Sass de Bootstrap
- Créer une **présentation esthétique** du site (vous avez carte blanche, tant que toutes les fonctionnalités sont présentes et qu'elles sont ergonomiques).
- Il faut ajouter dynamiquement de **l'animation** sur l'apparition des cards et l'ouverture de la fenêtre modale.
- Vous devez utiliser au moins une **fonction anonyme** ou **fonction fléchée**.



- Utiliser un **attribut HTML personnalisé** (Bouton supprimer dans chaque card <button **data-id**="task01">Supprimer</button>)

Fonctionnalités

- **Chargement** de nos données de tâches (données fournies constante **DATA_TACHES**) dans la DataTable de la librairie JavaScript Google charts.
- Affichage d'un **diagramme de Gantt** pour illustrer les tâches dans le temps (librairie JavaScript **Google charts**)
- **Affichage de chaque tâche** sous forme de composant **Card** de Bootstrap pour indiquer tous les détails de chaque tâche.
- **Modification d'une tâche** en sélectionnant une tâche dans le graphique (Événement « SELECT ». Voir la gestion des événements de Google Charts, voir la gestion des événements dans la documentation). Pour afficher les détails de tâche dans un formulaire afin de les modifier, on aura recourt à un **composant Modal** (fenêtre modale) de **Bootstrap**. Il existe des méthodes **show** et **hide** pour en gérer l'affichage.
- **Suppression** d'une tâche
- **Calcul du temps passé sur l'avancement d'une tâche.** Lorsqu'on modifie une tâche, on a accès à une minuterie (dans le formulaire d'édition) qui permettra de calculer le temps passé sur la réalisation d'une tâche. Pour simuler la réalité, disons qu'une seconde représente une journée. En démarrant la minuterie via un bouton, un champ de formulaire récupèrera le nombre de jours passés sur la tâche. Un autre bouton permettra d'ajuster le % d'avancement de la tâche.

Bibliothèque logicielle JavaScript

Dans ce TP

- Google Charts (graphiques pour visualiser des données)

Outils utiles pour notre travail :

- Graphique utilisé > Diagramme de Gantt (documentation). Il serait bien de démarrer vos tests en prenant une copie du code de la documentation et l'adapter à notre projet.
- Le **DataTable** (structure de données pour les Google charts, voir la documentation).
  - `dt = new google.visualization.DataTable();`
    - dt.**setValue** / dt.**getValue**
    - dt**.addColumn** / dt**.addRow**
    - dt.**getNumberOfColumns**
    - dt.**getNumberOfRows**
    - dt.removeRow
- Le **graphique de Gantt** (documentation).
  - `gantt = new google.visualization.Gantt(document.getElementById("chart_div"));`
    - gantt**.draw** : Rafraîchir l'affichage du diagramme
    - gantt**.getSelection** : Obtenir la rangée sélectionnée dans le graphique

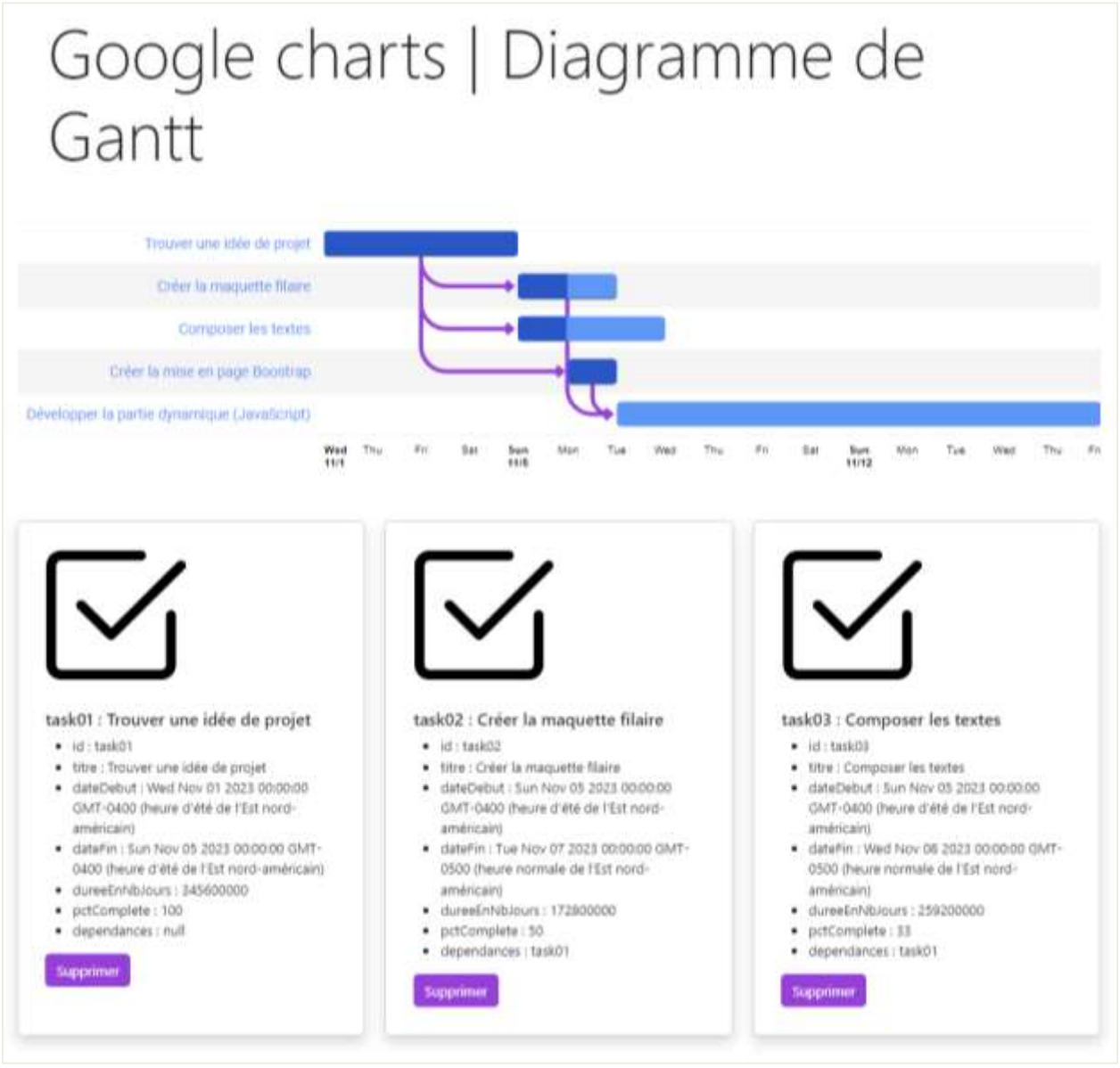
Page d'accueil

Voici les éléments que nous devrons retrouver de façon statique :

- Le header
  - Titre de contenu : « Gestionnaire de tâches »
- Section Diagramme
- Section Tâches (cards)
- Le footer
  - Comme dans le TP3 afin de vous identifier

Figure 1 - Voir une vidéo de démonstration

Design graphique



Liste des tâches / fonctions à créer

Vous devez développer les fonctions nommées suivantes en respectant leur nom exact et les paramètres.

1. Créer la structure HTML statique en Bootstrap
2. Fonctions propres à l'application (**javascript.js**) :
    a. `initialisation()`. Fonction exécutée après que le DOM HTML soit chargé. Dans cette fonction, on va en profiter pour lancer les méthodes de base de google.charts (soit load et setOnLoadCallback qui va appeler notre fonction `chargerEtAfficherDonneesDiagrammeEtCards` une fois que la librairie JavaScript de Google est prêt). Voir la documentation.
    b. `afficherCardsTaches()`. Affiche toutes les tâches sous forme de card. On vide la section des cards, puis on recréer l'affichage. On appelle la fonction utilitaire creerCard. Le titre est le id et titre de la tâche, le contenu est le reste des propriétés d'une tâche.
    c. `chargerEtAfficherDonneesDiagrammeEtCards()`. Fonction qui met en place le visuel de la page. On charge les données pour le graphique, on instancie le graphique de Google, on le dessine, puis on affiche les cards en appelant la fonction afficherCardsTaches. Cette fonction est appelée lorsque la bibliothèque de base de Google est complètement chargée.
    d. `creerDonneesPourGraphique()`. Permet de bâtir le DataTable que Google a besoin à partir de notre base de données (DATA_TACHES). Les méthodes addColumn et addRow du DataTable seront utiles pour cette fonction.
    e. `recupererTacheSelectionneeDansDiagrammeDeGantt()`. Fonction exécutée (déclenchée par l'événement SELECT du diagramme de Gantt) lorsqu'on sélectionne une tâche dans le diagramme. Cette fonction permet d'afficher les données dans une formulaire intégré à une fenêtre modale pour les éditer.
    f. `calculerAvancement`. Cette fonction est appelée par une minuterie aux secondes. Elle compte les secondes et les affiche dans le champ Réalisation. On simule ici qu'une seconde équivaut à 1 journée de travail. On calcule ensuite le % d'avancement (nb jours réalisés sur le nombre de jours estimé.). On anime finalement une progressBar (composant Bootstrap) en appliquant dynamiquement un style (width: X%)
    g. `arreterMinuterie`. Arrête la minuterie.
    h. `sauvegarderChangementsTache()`. Mettre à jour les données du DataTable du diagramme de Gantt. Les fonctions setValue (méthode du DataTable) et daysToMilliseconds vous seront utiles.
    i. `verifierSiDependanceExiste(pIdTache)`. Vérifie si la tâche fait partie des dépendances d'autres tâches. Si le ID de la tâche est

présent dans le tableau dependancies d'une autre tâche, il y a alors dépendance.

j. supprimerTache(e). Supprimer une tâche dont le Id est spécifié dans un attribut HTML personnalisé « data-id » dans les données du graphique et réafficher les cards, puis redessiner le graphique. On peut accéder à l'élément card via le paramètre e (qui est l'événement déclencheur). On ne peut pas supprimer une tâche lorsqu'il y a des dépendance (utilisation de la fonction verifierSiDependanceExiste)

3. Fonctions utilitaires (**fonctions-utilitaires.js**) :
   a. convertirJoursEnMillisecondes(pNbJours). Fonction qui convertit un nombre de jours en millisecondes et le retourne.
   b. convertirMillisecondesEnJours(pNbMillisecondes). Fonction qui convertit un nombre de millisecondes en jours et le retourne.
   c. creerCard(pImage, pTitre, pDescription, pEstAvecBouton, pElementHTMLBouton). Créer la structure HTML pour afficher une card à partir des paramètres pour personnaliser l'image, le titre et la description, si on veut afficher un bouton ou non et, le cas échéant le HTML composant ce bouton.
4. Événements à prévoir :
   a. « click » : Clic du bouton Sauvegarder les changements (lorsqu'on modifie les données d'une tâche).
   b. « DOMContentLoaded » : Lancement de la fonction d'initialisation lorsque le DOM est prêt événement
   c. « resize » : Rafraîchissement du graphique au redimensionnement de la fenêtre (objet window)

Répartition des tâches

****** Travail en équipe ******

Il est essentiel que les deux membres de l'équipe collaborent étroitement et soient de bonne foi pour la réalisation du travail en équipe. Si toutefois un étudiant détecterait un problème important avec son coéquipier lors de la réalisation du travail, il est de sa responsabilité d'en aviser le professeur le plus rapidement possible et non pas seulement lors de la remise

---

Étudiant 1

- ***initialisation***
- *creerDonneesPourGraphique*
- *chargerEtAfficherDonneesDiagrammeEtCards*
- *afficherCardsTaches*
- ***supprimerTache*** *(utilisation d'un attribut HMTL personnalisé)*
- *creerCard*
- *verifierSiDependanceExiste*
- *Rafraîchir l'affichage du diagramme lors de l'événement « resize » de la fenêtre +* ***Utilisation d'une fonction anonyme*** *ou fléchée*
- *Créer une animation sur l'affichage des cards*

Étudiant 2

- ***initialisation***
- *Utiliser une fenêtre modale Bootstrap 5 qui contient un formulaire pour éditer les données d'une tâche*
- *recupererTacheSelectionneeDansDiagrammeDeGantt*
- *sauvegarderChangementsTache +* ***Utilisation d'une fonction anonyme ou fléchée***
- ***supprimerTache*** *(utilisation d'un attribut HMTL personnalisé)*
- *calculerAvancement*
- *arreterMinuterie*
- *ajusterAvancementTacheSelonMinuterie*
- *convertirJoursEnMillisecondes*
- *convertirMillisecondesEnJours*

Grille d'évaluation

Éléments	Points totaux
Bon fonctionnement de l'application	**10**

JavaScript : Intégrer et utiliser une bibliothèque JavaScript
• Bon fonctionnement de la bibliothèque Google Charts
• Utilisation des méthodes pour manipuler le DATATABLE
  ○ dt = **new google.visualization.DataTable();**
  ○ dt.**setValue** / dt.**getValue**
  ○ dt**.addColumn** / dt**.addRow**
  ○ dt.**getNumberOfColumns**
  ○ dt.**getNumberOfRows**
  ○ dt.removeRow
• Utilisation des méthodes pour manipuler le diagramme de Gantt
  ○ gantt**.draw** : Rafraîchir l'affichage du diagramme
  ○ gantt**.getSelection** : Obtenir la rangée sélectionnée dans le graphique
	10

Javascript : Attributs HTML personnalisés • Utilisation d'un attribut HTML personnalisé sur le bouton supprimé • Il permet d'injecter une donnée dans le HTML et de la manipuler par JavaScript (id de la tâche)	**10**
Javascript : Utiliser des fonctions de minuterie • Début d'une minuterie pour calculer le temps • Arrêt de la minuterie • Utilisation de la donnée pour calculer un nombre de jours	**10**
JavaScript : Animation dynamique • Application d'une animation dynamiquement sur le DOM HTML	**10**
JavaScript : Utilisation d'une fonction anonyme ou fléchée • Utilisation d'une fonction anonyme pour simplifier certains traitement ou pour pouvoir passer des paramètres dans les fonctions appelées via un événement	**10**
Validation du code et de l'accessibilité • HTML 5 • WGAG 2.1 AA	**5**
Git et GitLab • Configuration du dépôt • Fichiers ignorés • Commit fréquents • Qualité des messages • 2x Issues réglées, avec #commit • Bonne participation des 2 coéquipiers	**5**

Sass avec Bootstrap : • Utilisation correcte de Saas avec Bootstrap. • Adapter les couleurs et la police par défaut selon les normes de design • Utilisation des directives @extend pour pouvoir mettre les classes de Bootstrap liées aux mises en forme directement dans le fichier Sass. • Fichier Sass très bien structuré (sections + commentaires).	**10**
Qualité du code JavaScript : • Code clair et facile à comprendre. • Code bien présenté. • Respect de la syntaxe ESLint utilisée dans le cadre du cours. • Constante globale (« data ») non modifiée ajoutée en /*global*/ • Respect ou peaufinement du découpage en fonctions JavaScript ayant des buts précis. • Bons commentaire JSDoc pour les fonctions JavaScript.	**10**
Design graphique • Qualité du design • Respect des bonnes pratiques • Mariage des polices Google et génération de palette de couleurs harmonieuses • etc…	**10**

Pénalités

- Retard :
- Présentation du code (décalage, disposition et sauts de ligne) :
- Structure de fichiers (bons dossiers, fichiers aux bons endroits, noms des fichiers significatifs, pas de fichiers inutiles, etc.) :
- Remise incorrecte :
- Qualité du français (structure des phrases et fautes) :
- Autres :