

# IFT1015 - Travail Pratique 2

## Sondage de Dates

Concepts appliqués : boucles, tableaux, fonctions, décomposition fonctionnelle, traitement d'événements, programmation web

---

## 1 Contexte

Pour ce deuxième travail pratique, vous devrez *compléter* le code d'une application web (client et serveur) utilisant `node.js`.

L'application web que vous aurez à compléter est une application dans le style du site `doodle.com`, qui permet de planifier des réunions et autres activités aux moments qui conviennent le mieux à tous les participants.

## 2 Structure du code

La logique de base du serveur vous est fournie, vous aurez la tâche de compléter la page de sondage ainsi que la page d'affichage des résultats.

Les fichiers fournis sont les suivants :

```
tp2/
|-- index.js ----- Code du serveur
|
|-- public/ ----- Dossier contenant les fichiers accessibles via le web
|   |-- calendar.js ----- Fichier contenant le code javascript du client
|   |-- pickmeup.* ----- Utilitaire pour sélectionner des dates
|   |-- calendar.css ----- Fichier de style pour la page de choix
|   |-- index.css ----- Fichier de style pour l'accueil
|   |-- results.css ----- Fichier de style pour la page de résultats
|
|-- template/ ----- Dossier contenant les fichiers HTML
    |-- index.html ----- Page d'accueil (nouveau sondage)
    |-- calendar.html ----- Page pour la sélection de dates
    |-- results.html ----- Page de visualisation des réponses
    |-- error404.html ----- Page d'erreur
```

Les seuls fichiers que vous aurez à modifier sont `index.js` et `public/calendar.js` (à moins de faire les bonus, voir les détails plus bas).

### 3 Tester

Pour lancer le serveur, exécutez la commande suivante depuis le dossier où vous avez extrait les fichiers du tp :

```
nodejs index.js
```

Une fois le serveur lancé, ouvrez la page `http://localhost:1337/` dans votre navigateur.

## 4 Application finale

L'application finale doit permettre les actions suivantes :

### 4.1 Créer un sondage

URL: `http://localhost:1337/index.html`

**crée une réunion, seulement si les info sont correctes**

### Créer un sondage

Titre:

Identifiant:  (?)


Date de début:

Date de fin:

Heure de début:

Heure de fin:

**Ça dérange pas que si on ferme le serveur, on perd les données**



**créerSondage est une procédure**

Figure 1: Création d'un nouveau sondage

Cette page est déjà présente. Lorsqu'un utilisateur clique sur le bouton *Créer !*, la fonction `créerSondage()` du fichier `index.js` est appelée du côté du serveur, avec en paramètres les différentes valeurs entrées dans le formulaire.

Vous devrez remplir cette fonction pour ajouter un nouveau sondage en mémoire. **Réfléchissez à la façon dont vous gardez les sondages en mémoire.** Vous n'avez pas besoin d'utiliser des fichiers (autrement dit, les sondages créés peuvent être perdus dès qu'on ferme le serveur), mais vous pouvez en utiliser si vous le voulez.

Avant d'enregistrer le nouveau sondage, assurez-vous que les informations sont valides :

- L'identifiant `id` doit contenir seulement des chiffres, des lettres (majuscules/minuscules) et des tirets – **tester avec les accents**
- L'heure de début doit être plus petite ou égale à l'heure de fin
- La date de début doit être plus petite ou égale à la date de fin
- Le nombre de jour maximum permis pour un sondage est 30

La fonction `creerSondage` doit retourner `true` si le sondage a été créé correctement, ou `false` si les informations sont invalides.

## 4.2 Répondre à un sondage

URL: `http://localhost:1337/{identifiant de sondage}`

**Réunion très importante**

	18 Nov	19 Nov	20 Nov	21 Nov	22 Nov	23 Nov
7h						
8h						
9h						
10h						
11h						
12h						
13h						
14h						
15h						
16h						
17h						

Nom:

Partagez ce sondage en utilisant le lien suivant : <http://localhost:1337/superreunion>

**Une fois que l'utilisateur clique sur participer... Il retourne un tableau avec les possibilités**

Figure 2: Plages-horaires possibles pour un sondage donné – on doit pouvoir sélectionner et désélectionner des disponibilités en cliquant et déplaçant la souris sur les cases

Lorsqu'on tente de répondre à un sondage, la fonction `getCalendar()` du fichier `index.js` est appelée avec en paramètre l'identifiant du sondage demandé.

La fonction devrait retourner le contenu de la page `template/calendar.html`, avec les `{{balises}}` remplacées par le contenu approprié :

- `{{titre}}` correspond au titre du sondage donné lors de la création
- `{{table}}` correspond à la grille affichant les plages horaires et permettant d'entrer ses disponibilités
- `{{url}}` correspond à l'URL de la page actuelle

#### 4.2.1 Tableau de plages horaires

Vous pouvez créer le tableau des plages horaires possibles pour un sondage donné en utilisant une `<table>` HTML dans ce format (exemple pour obtenir le tableau montré à la *Figure 2*) :

```
<table id="calendrier"
  onmousedown="onClick(event)"
  onmouseover="onMove(event)"
  data-nbjours="6"
  data-nbheures="11">
  <!-- En-tête -->
  <tr>
    <th></th>
    <th>18 Nov</th>
    <th>19 Nov</th>
    <th>20 Nov</th>
    ...
  </tr>
  <tr>
    <th>7h</th>
    <td id="0-0"></td>
    <td id="1-0"></td>
    <td id="2-0"></td>
    ...
  </tr>
  <tr>
    <th>8h</th>
    <td id="0-1"></td>
    <td id="1-1"></td>
    <td id="2-1"></td>
    ...
  </tr>
  <tr>
    <th>9h</th>
    <td id="0-2"></td>
    <td id="1-2"></td>
    <td id="2-2"></td>
    ...
  </tr>
  ...
</table>
```

**Comment on sait où ça commence? ex. nov ou 18 au 30? Comment l'info est transférée?**

**On va pouvoir extraire le id avec onClick**

- L'utilisation des attributs `onmousedown` et `onmouseover` permettra d'appeler automatiquement les fonctions `onMove` et `onClick` du fichier `public/calendar.js` lorsque l'utilisateur survole ou clique sur le calendrier, respectivement
- L'id des cellules du tableau vous permettra d'identifier les cellules affectées par un événement `click` ou `move`
- Les valeurs des attributs `data-nbjours` et `data-nbheures` seront accessibles dans `public/calendar.js` en utilisant :

```
var cal = document.getElementById("calendrier");
var nbHeures = cal.dataset.nbheures;
var nbJours = cal.dataset.nbjours;
```

#### 4.2.2 Soumettre le formulaire

Lorsqu'on clique sur le bouton *Participer*, la fonction `compacterDisponibilites()` est appelée. Cette fonction doit retourner une série de zéros et de uns qui encode les disponibilités sélectionnées, de gauche à droite et de haut en bas. C'est cette valeur qui sera envoyée au serveur.

	18 Nov	19 Nov	20 Nov
12h	✓ 1	✓ 1	0
13h	0	✓ 1	0
14h	0	✓ 1	0

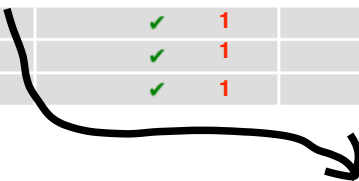


Figure 3: `compacterDisponibilites()` => retourne 110010010 ici

#### 4.2.3 Enregistrer un nouveau participant

Lorsque le formulaire est soumis, la fonction `ajouterParticipant()` est appelée du côté du serveur avec en paramètre l'identifiant du sondage, le nom de la personne qui a répondu et les disponibilités entrées.

Vous devez stocker les disponibilités entrées quelque part pour pouvoir les afficher sur la prochaine page.

### 4.3 Consulter les résultats d'un sondage

URL: `http://localhost:1337/{identifiant de sondage}/results`

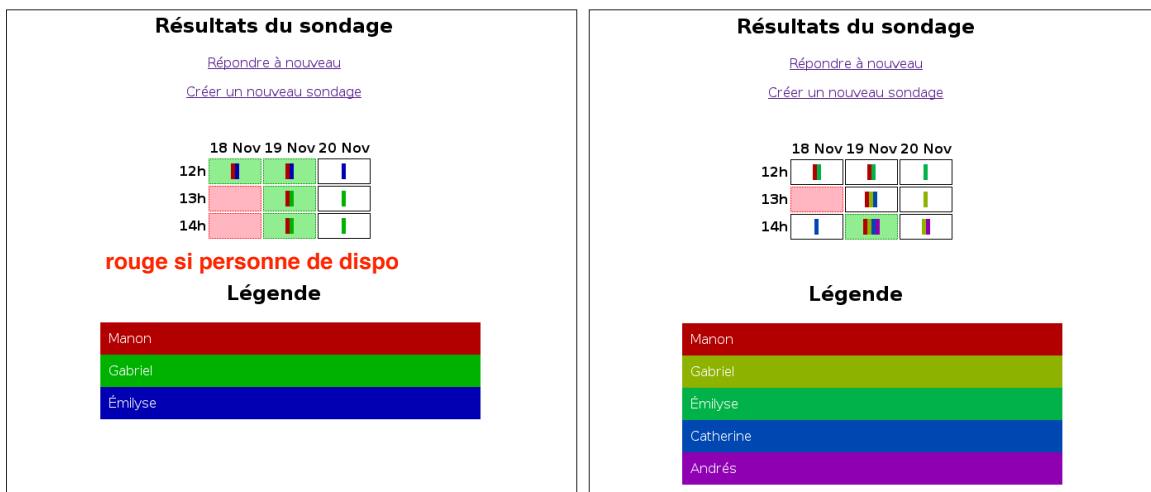


Figure 4: Exemples de page de résultats avec 3 et 5 participants

La table de résultats est une grille indiquant une petite barre colorée pour chaque disponibilité inscrite dans chaque case. Chaque participant est associé à une couleur, les couleurs sont générées automatiquement. Dans l'exemple de la *Figure 4*, Manon est disponible les 18 novembre à 12h et le 19 novembre de 12h à 14h. Il y a donc une barre rouge dans chacune de ces cases.

Les cases correspondant au plus grand nombre de personnes disponibles sont en vert, tandis que les cases correspondant au plus petit nombre de personnes disponibles sont en rouge.

Lorsqu'on veut afficher les résultats d'un sondage, la fonction `getResults()` du fichier `index.js` est appelée avec en paramètre l'identifiant du sondage demandé.

La fonction devrait retourner le contenu de la page `template/results.html`, avec les `{{balises}}` remplacées par le contenu approprié :

- `{{url}}` et `{{titre}}` : même chose que sur l
- `{{table}}` correspond à la grille affichant les résultats du sondage
- `{{legende}}` correspond à une explication du code couleur utilisé

#### 4.3.1 <table> de résultats

La <table> représentée dans le premier panneau de la *Figure 4* pourrait être générée comme suit :

```
<table>
  <tr>
    <th></th> <th>18 Nov</th> <th>19 Nov</th> <th>20 Nov</th>
  </tr>
  <tr>
    <th>12h</th>
    <td class="max">
      <span style="background-color: #b20000; color:#b20000">.</span>
      <span style="background-color: #0000b2; color:#0000b2">.</span>
    </td>
    <td class="max">
      <span style="background-color: #b20000; color:#b20000">.</span>
      <span style="background-color: #0000b2; color:#0000b2">.</span>
    </td>
    <td>
      <span style="background-color: #0000b2; color:#0000b2">.</span>
    </td>
  </tr>
  <tr>
    <th>13h</th>
    <td class="min"></td>
    <td class="max">
      <span style="background-color: #b20000; color:#b20000">.</span>
      <span style="background-color: #00b200; color:#00b200">.</span>
    </td>
    <td>
      <span style="background-color: #00b200; color:#00b200">.</span>
    </td>
  </tr>
  ...
</table>
```

Une barre colorée peut être générée avec le code HTML suivant :

```
<span style="background-color: {couleur}; color: {couleur}">.</span>
```

Où {couleur} correspond à la couleur du participant.

Les classes CSS `min` et `max` sont fournies et permettent de colorier les cases qui correspondent au plus petit et au plus grand nombre de disponibilités en rouge et en vert, respectivement.

### 4.3.2 Légende

La légende peut être générée avec une liste non ordonnée. La légende dans le deuxième panneau de la *Figure 4* aurait l'air de :

```
<ul>
  <li style="background-color: #b20000">Manon</li>
  <li style="background-color: #8eb200">Gabriel</li>
  <li style="background-color: #00b247">Émilys</li>
  <li style="background-color: #0047b2">Catherine</li>
  <li style="background-color: #8e00b2">Andrés</li>
</ul>
```

### 4.3.3 Génération de couleur

Pour générer les couleurs utilisées par chaque participant, on peut utiliser la formule suivante :

$$\text{teinte} = \frac{\text{numéro de couleur}}{\text{nb total de couleurs}} * 360$$

$$h = \frac{\text{teinte}}{60}, c = 0.7$$

$$x = c * (1 - |h \% 2 - 1|)$$

$$(r, g, b) = \begin{cases} (c, x, 0) & \text{si } \lfloor h \rfloor = 0 \\ (x, c, 0) & \text{si } \lfloor h \rfloor = 1 \\ (0, c, x) & \text{si } \lfloor h \rfloor = 2 \\ (0, x, c) & \text{si } \lfloor h \rfloor = 3 \\ (x, 0, c) & \text{si } \lfloor h \rfloor = 4 \\ (c, 0, x) & \text{si } \lfloor h \rfloor = 5 \\ (0, 0, 0) & \text{sinon} \end{cases}$$

$(r, g, b)$  correspond à la couleur voulue. En HTML, la couleur peut être représentée par `#RRGGBB`, où `RR` est la quantité de rouge (de 0 à 255) au format hexadécimal (donc de 00 à FF), `GG` est la quantité de vert et `BB` est la quantité de bleu.

Vous devrez remplir la fonction `genColor(i, total)` qui retourne la  $i^{\text{ème}}$  couleur sur un total de `total` au format HTML hexadécimal.

## 5 Quelques conseils...

Ce travail pratique n'est pas *difficile*, mais il comporte *beaucoup d'étapes*.

Il est *complexe*, mais pas *difficile*. Allez-y une étape à la fois, dans l'ordre.

Avant de commencer, jouez avec le serveur et assurez-vous de comprendre ce qui se passe. Utilisez des `console.log()` un peu partout pour afficher le contenu des variables et des paramètres passés aux fonctions pour visualiser l'état du serveur.

Réfléchissez bien à la façon dont vous souhaitez stocker les sondages et les disponibilités des participants. Pensez à ce qui serait le plus approprié (tableaux ? enregistrements ? tableaux d'enregistrements ? autres ?)

## 6 Bonus (+10%)

### 6.1 (+6%) Failles de sécurité

Si vous faites le travail tel que demandé, votre serveur final ne sera pas parfait... On pourrait par exemple parler du Cross-site scripting (XSS). Corrigez les problèmes de XSS sur le serveur pour +3% bonus. Pour plus d'informations, Wikipédia et Google sont vos amis : [https://fr.wikipedia.org/wiki/Cross-site\\_scripting](https://fr.wikipedia.org/wiki/Cross-site_scripting)

+3% pour chaque faille de sécurité supplémentaire que vous trouverez et corrigerez. Il y a au moins une deuxième faille de sécurité connue, mais vous pourriez en trouver d'autres...

Assurez-vous de détailler toutes les corrections de sécurité que vous faites dans un fichier nommé README.txt que vous remettrez avec le reste du code. Vous devez expliquer chaque faille trouvée, comment l'exploiter, et quel bout de code ajouté permet de la corriger.

Notez finalement que si *votre propre code* est la source d'une faille de sécurité, vous n'aurez pas de bonus simplement pour l'avoir corrigée...

### 6.2 (+4%) Message d'erreur lors de la création d'un sondage

Dans la version initiale du serveur, le message d'erreur est loin d'être convivial :

**Un des points bonus plus accessibles**

#### Créer un sondage

Erreur : assurez-vous d'entrer des données valides.

Figure 5: Message d'erreur de base pour les formulaires invalides

Corrigez le message d'erreur de façon à expliquer de façon compréhensible et en mots simples la source de l'erreur, par exemple :

- L'identifiant de sondage correspond à un sondage existant
- L'heure de fin doit être après l'heure de début
- La durée maximale d'un sondage est de 30 jours
- etc



## 7 Évaluation

- Ce travail compte pour 15 points dans la note finale du cours. Vous devez faire le travail en équipes de 2 personnes. Indiquez vos noms clairement dans les commentaires au début des fichiers `index.js` et `public/calendar.js`.

- **Remettez tout le code nécessaire pour faire rouler l'application, dans une archive .zip.** Vous pouvez exécuter la commande :

```
zip -r remise.zip *
```

depuis le dossier qui contient le code du serveur pour créer l'archive.

- La remise doit se faire au plus tard **vendredi le 14 décembre à 23h55** sur le site Studium du cours.
- Voici les critères d'évaluation du travail : l'exactitude (respect de la spécification), l'élégance et la lisibilité du code, la présence de commentaires explicatifs, le choix des identificateurs, la décomposition fonctionnelle, le choix de tests unitaires pertinents et l'utilisation d'un français sans fautes. La performance de votre code doit être raisonnable. Chaque fonction devrait avoir un bref commentaire pour dire ce qu'elle fait, il devrait y avoir des lignes blanches pour que le code ne soit pas trop dense, les identificateurs doivent être bien choisis pour être compréhensibles et respecter le standard camelCase.