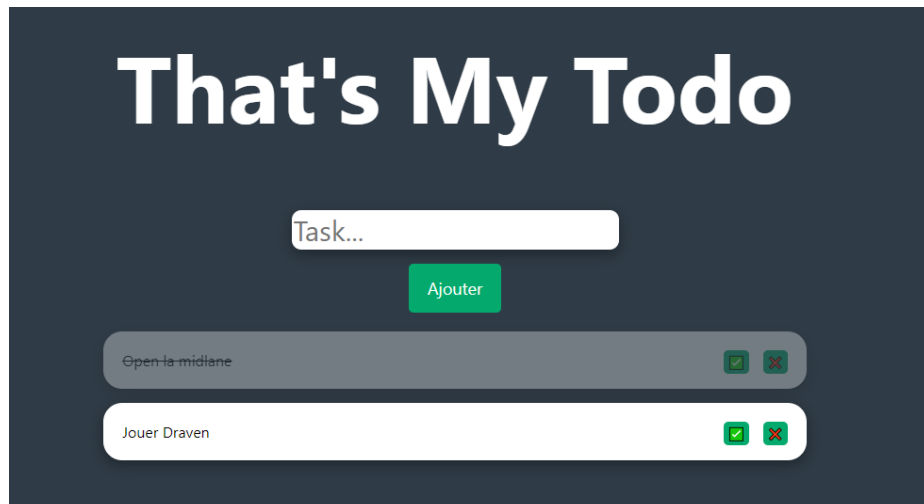


## Rapport Web

### Page d'accueil



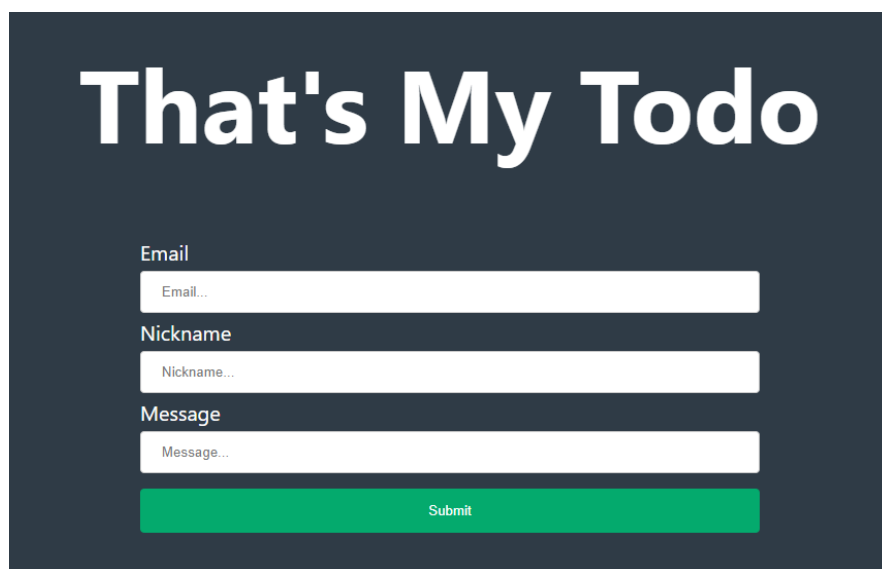
The screenshot shows the 'That's My Todo' application interface. At the top, the title 'That's My Todo' is displayed in large white font on a dark blue background. Below the title is a text input field labeled 'Task...' with a green 'Ajouter' button to its right. Underneath the input field is a list of tasks. The first task is 'Open la midlane', which is partially faded (opacity 0.33) and has a green checkmark and a red 'X' button to its right. The second task is 'Jouer Draven', which is not faded and also has a green checkmark and a red 'X' button to its right.

Je vous présente la page d'accueil de ma TODO List fait en HTML/CSS/JSS. Comme vous pouvez le voir nous pouvons écrire du texte dans le champ *Task...* puis ensuite nous pouvons cliquer *Ajouter* ou appuyer sur entrée pour ajouter une tâche.

Chaque tâche peut être marquée comme effectuée grâce au bouton vert, cela rendra l'opacité de la tâche à 0.33 et barrera le texte.

On peut également supprimer la tâche en appuyant sur le bouton rouge.

### Page de contact



The screenshot shows the 'That's My Todo' application contact page. At the top, the title 'That's My Todo' is displayed in large white font on a dark blue background. Below the title are three text input fields labeled 'Email', 'Nickname', and 'Message'. Each input field has a placeholder text 'Email...', 'Nickname...', and 'Message...' respectively. Below the input fields is a green 'Submit' button.

Maxence JUNG

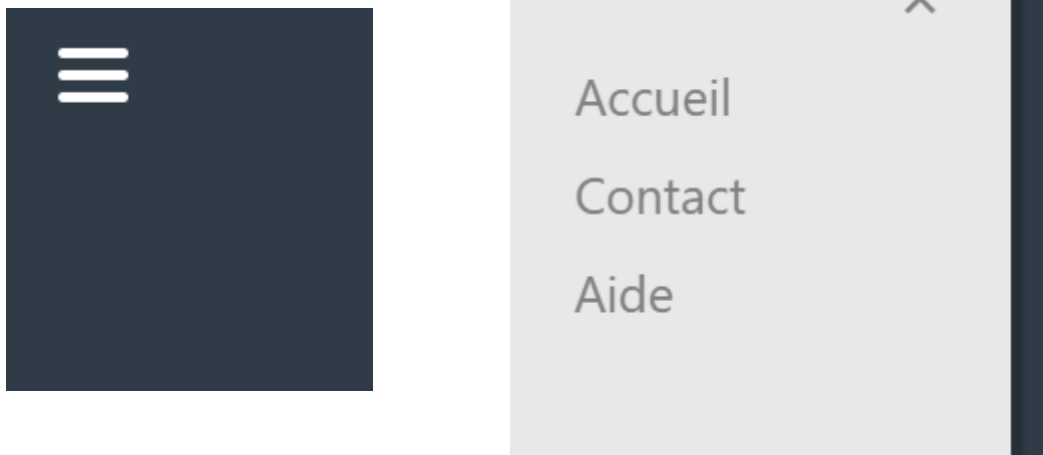
Ensuite nous avons la page de contact avec un formulaire classique contenant l'email, le pseudonyme et le message. Ici le formulaire ne fait rien je ne n'ai pas programmé de back-end.

## Page d'aide



Et finalement voici ma page d'aide, je n'ai pas eu beaucoup d'inspiration pour celle-ci.

## Menu Burger.



Voici mon menu de navigation. C'est ce qu'on appelle un menu burger. En cliquant le menu s'ouvre avec une animation.

## Programme Javascript

J'ai créé un fichier *script.js* pour gérer mes tâches composé de plusieurs fonctions. Je vais en expliquer le fonctionnement.

### - Load()

Tout d'abord le programme vérifie la clé "json" dans le localStorage, si il n'existe pas le programme va créer un JSON contenant un tableau nommé *task*.

Le format de chaque tâche est le suivant, il contient une clé *desc* avec la description de la tâches sous forme de chaînes de caractères ainsi qu'une clé *done* le booléen signifiant que la tâche est effectué.

Ensuite nous récupérons le JSON dans le localStorage puis nous parsons ce JSON car il est sous forme de chaîne de caractères.

Dans le cas où le JSON ne contiendrait aucune tâche, nous créerons un élément paragraphe avec le texte "Aucune tâche".

Sinon pour chaque tâche nous créerons une div contenant le texte de la tâche et les deux boutons.

La particularité des boutons ici est que nous déclarons un attribut onClick avec leur indice *i* dans le JSON. Cela nous permettra de marquer les tâches comme effectué ou de les supprimer plus facilement.

Dans le cas où le clé *done* nous modifierons le style du paragraphe en le barrant et nous modifierons l'opacité de la div à 0.33.

Puis nous ajouterons cette div nommé "task" à la div "tasks".

### - RemoveTask(index)

Tout d'abord nous récupérons le JSON contenu sous forme de chaîne de caractères dans le localStorage. Ensuite nous le parsons.

Grâce à l'indice *i* passé en argument pendant la fonction Load() nous avons juste à utiliser la méthode splice pour supprimer l'élément du JSON.

Finalement nous avons juste à remettre le contenu dans le localStorage. Vider la div tasks et rappeler la fonction Load() pour recharger les tâches.

### - taskDone(index)

Le fonctionnement est similaire à la fonction removeTask(index). Nous faisons exactement les mêmes étapes que précédemment sauf que au lieu de supprimer l'élément nous changeons sa propriété *done* booléenne à l'inverse de sa valeur actuelle.

Maxence JUNG

- `addTask()`

Pour ajouter une tâche, nous allons d'abord récupérer le texte de l'input. Puis comme précédemment nous récupérons le JSON depuis le `localStorage`. Nous ajoutons la tâche avec le texte et la clé *done* initialisé à `false` grâce à la méthode `push`.

Nous actualisons le `localStorage` puis nous vidons l'input ainsi que la div *tasks* afin de pouvoir rappeler la fonction `Load()` pour recharger les tâches.

- script Burger

Ce script permet tout simplement d'ouvrir et fermer le menu sur le côté en obtenant les éléments via les *id*.

## Conclusion

Je n'ai eu aucune difficulté à faire ce projet car j'ai déjà fait du web dans mes précédentes années universitaires. Je n'avais jamais utilisé le `localStorage` avant c'était une bonne expérience. Les prochains objectifs seraient de stocker les tâches avec un service de cloud puis avec un système de compte utilisateur, proposer de créer un compte et de pouvoir stocker ses tâches afin de pouvoir les avoir sur plusieurs appareils.