

Web avancé – L2 MIASHS

TD 06 - AJAX, encore et toujours !

Continuons à pratiquer les requêtes asynchrones vers un serveur distant. Cette fois-ci il n'est pas question d'aborder des films, mais bel et bien de faire davantage de requêtes vers une API publique dédiée aux chats 🐱.

Quelques petites différences toutefois avec le TD précédent : cette fois-ci nous allons utiliser un formulaire pour regrouper tous les choix utilisateurs de manière plus pratique. Bien entendu, comme nous gérons les requêtes par Javascript, nous pouvons déclencher l'événement de click de manière dissociée du formulaire. Il n'est donc pas obligatoire d'utiliser un bouton de type submit !

Objectif : Pratiquer les requêtes asynchrones

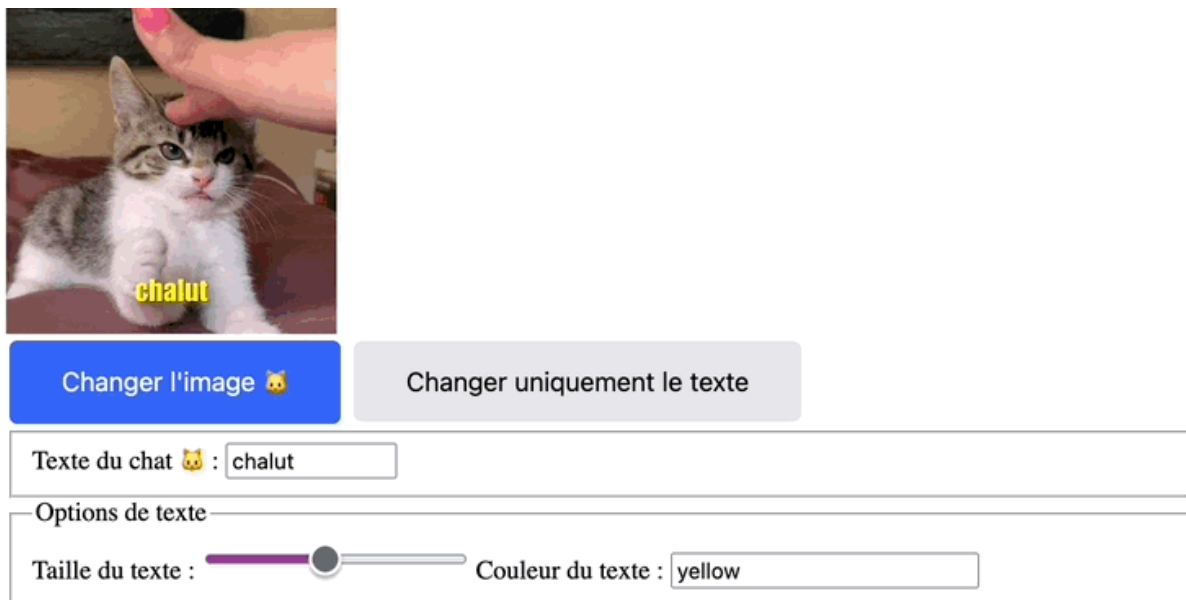
Cible : les étudiants connaissant déjà les requêtes asynchrones en Javascript ainsi que tous les fondamentaux de JS. Le CSS et le HTML sont considérés comme déjà maîtrisés.

Durée : très variable. Quoi qu'il en soit, essayez de vraiment finir ce TD ! Même chez vous.

Chaque exercice se distingue par ce logo 🐱 !

Dans ce TD, le code est intégré sous forme d'image afin que vous ne puissiez pas simplement le copier-coller.

Au terme de ce TD vous devriez obtenir le résultat suivant :



Affichage d'une image de chat aléatoire

Commençons par la première étape : l'affichage d'une image de chat aléatoire. Nous devons pour cela créer le champ de réception de l'image ainsi que le bouton. Puis gérer le click sur le bouton qui va déclencher une requête asynchrone vers le serveur distant.




Dans ce TD, nous ne nous concentrerons pas sur le CSS, le fichier vous est déjà fourni et c'est à vous d'en utiliser les classes prédéfinies.

D'ailleurs, vous trouverez le dossier de départ au lien suivant :

<https://arche.univ-lorraine.fr/mod/resource/view.php?id=1884678>

Détaillons les étapes nécessaires pour créer le HTML nécessaire dans **main.html** :



- Mettez un titre d'onglet égal à “ Le monde des chats  !”
- Liez les fichiers CSS et Javascript au fichier HTML
- Créez une balise **div** ayant l'identifiant “resultView”. Elle nous servira de point d'ancrage pour insérer l'image du chat. D'ailleurs, le fichier CSS y force une taille de 200 pixels de haut
- Créez une balise **button** ayant les classes “button” et “primary”, ainsi que le texte “Changer l'image ”

Maintenant que tout est prêt, passons au Javascript nécessaire.

Pour cela, dans **script.js** :



- Créez une fonction asynchrone qui
 - vide le contenu HTML de la balise ayant l'identifiant “resultView” à l'aide de innerHTML
 - crée une balise **img** ayant pour source l'image GIF de chargement contenu dans le dossier img, puis ajoute cette balise à la balise ayant l'identifiant “resultView” (cela nous permet d'afficher une animation de chargement le temps d'obtenir la réponse du serveur)
 - utilise fetch() combiné à await pour chercher des informations à l'URL <https://cataas.com/cat>
 - Attention ! Il est important de préciser que l'on souhaite des données de type de JSON. Pour cela, utilisez le deuxième argument de la fonction fetch(url, options) qui est un tableau associatif de différentes options. Notamment, nous souhaitons indiquer les entêtes (headers) par un sous tableau associatif ayant une clé “headers” associée à la valeur “application/json”
 - utilise await pour transformer la réponse du serveur distant en objet json à l'aide de la méthode .json()
 - vérifie si le statut de la réponse est ok
 - Si la réponse est ok, utilise le json obtenu pour obtenir l'identifiant du chat aléatoirement. Cet identifiant est également l'identifiant de son

image. Par exemple, si l'identifiant est "chatmignon" alors l'URL deviendra "https://cataas.com/cat/chatmignon"

- Si la réponse n'est pas ok, sauvegarde l'URL en tant que :
"https://cdn.pixabay.com/photo/2017/02/12/21/29/false-2061132_640.png"
- utilise l'url obtenue pour créer une balise img ayant pour source cette URL et un [attribut height](#) d'une valeur de 200. Puis, l'ajoute au DOM par le biais de la balise ayant l'identifiant "resultView".
- Maintenant, attachez au bouton une [écoute de l'événement](#) click pour que, lors d'un clic, le bouton active la fonction asynchrone servant à requêter l'image et à l'afficher.

Et voilà ! Vous devriez pouvoir déjà voir le résultat en cliquant sur le bouton

Changer l'image 🐱

Texte personnalisé par le formulaire

Bon, malgré tout ça, ce n'est pas fini 😊! Comme vous le voyez dans le résultat attendu, nous devrions également pouvoir demander à l'API de nous renvoyer une image avec du texte personnalisé. Pour cela, il convient de bien regarder de nouveau la documentation de l'API cataas : <https://cataas.com/doc.html>

Si l'on en croit cette documentation, nous devons donc fournir le texte par complétion de l'URL. Ainsi, pour l'image ayant l'identifiant "BYAFp9AIPCIDCKgf", nous envoyions précédemment <https://cataas.com/cat/BYAFp9AIPCIDCKgf> tandis que, cette fois-ci, nous devons obtenir par exemple : <https://cataas.com/cat/BYAFp9AIPCIDCKgf/says/chalut>

À partir de la documentation et de cet exemple nous devons donc ajouter du contenu HTML et compléter notre fonction asynchrone.

Pour ce faire, dans **main.html** :



- créez une balise **form** ayant l'identifiant "miaouForm"
- dans cette balise **form**, créez une balise **fieldset** qui contient
 - une balise **label** ayant un attribut for avec valeur "text". Cette balise contient le texte "Texte du chat 🐱."
 - une balise **input** ayant l'identifiant "catText", de type text et un attribut "name" avec pour valeur "text". Pour rappel les balises **label** et **input** sont liées via le couple d'attributs "for" et "name" devant alors avoir la même valeur (ici "text").

Vous vous demandez sûrement ce qu'est cette balise fieldset ? C'est une balise d'HTML 5 qui permet d'avoir un petit peu de style par défaut pour des sections d'un formulaire.

Regardez donc la doc : <https://developer.mozilla.org/fr/docs/Web/HTML/Element/fieldset>

Passons maintenant au javascript, donc dans **script.js** nous devons modifier la fonction asynchrone et y ajouter la récupération des données contenues dans le formulaire. Pour cela :



- récupérez la balise ayant l'identifiant "miaouForm"
- créez une encapsulation des données du formulaire à l'aide d'une nouvelle instance de la classe `FormData` comme visible ici :

```
let formData = new FormData(form);
```

- transformez cette instance en un objet fait de clés et de valeurs... donc un tableau associatif bien plus simple à utiliser et à visualiser. Pour cela, il faut utiliser une méthode statique de la classe `Object`, la méthode `fromEntries()`. Comme ceci :

```
let userChoices = Object.fromEntries(formData);
```

- utilisez cette variable `userChoices` pour récupérer ce que l'utilisateur a choisi.
- ajoutez la transformation de l'URL de requête si l'utilisateur a bien écrit quelque chose. S'il a bien écrit un texte, ajoutez `/says/` suivi du texte du formulaire.

Contemplez le résultat. Vous devriez pouvoir cliquer sur le bouton pour obtenir une nouvelle image, et si vous indiquez un texte, une nouvelle image aléatoire est alors demandée avec du texte. Incroyable n'est-ce pas ?

N'est-ce pas ?!!

MAIS !!! Car il y a toujours un "mais" !

Qu'en est-il du fait de pouvoir demander la même image avec un nouveau texte ? Vous voyez que c'est effectivement le cas dans le résultat attendu mais pas dans ce que vous avez actuellement. Mince alors ! 😞

Pour cela il faut :



- Créer un nouveau bouton dans `main.html` aux côtés du précédent
 - lui donner la classe "button"
 - lui donner le texte "Changer uniquement le texte"
- Ajouter un écouteur d'événement sur ce bouton qui déclenche la fonction asynchrone sur un click, mais ! avec un argument booléen indiquant s'il faut demander une nouvelle image à partir du même ID ou demander un nouveau chat aléatoire
- Adapter la fonction asynchrone pour qu'elle puisse accepter un argument conditionnant la demande, ou non, d'un nouveau chat aléatoire
 - 💡 pour cela, je vous conseille également de sauvegarder l'ID de l'image en cours en tant que variable globale 😊

Cela fonctionne ? Vous devriez pouvoir changer le texte sans changer le chat. Ainsi, vous pouvez adapter ce que dit le chat en fonction de vos préférences vis-à-vis de l'image. Génial n'est-ce pas ? 😊

Ceci est désormais possible :



Pour respectivement, les URL suivantes utilisées par fonction asynchrone :

<https://cataas.com/cat/BYAFp9AIPCIDCKgf/says/chalut>

<https://cataas.com/cat/BYAFp9AIPCIDCKgf/says/miaou>

Style du texte par le formulaire

Tout semble correct et pratique. Il est cependant dommage de ne pas pouvoir changer le style du texte. Vous devez donc compléter le formulaire HTML et la méthode asynchrone afin de pouvoir indiquer la taille de la police et la couleur du texte.

Attention, cette fois-ci il s'agit de paramètres de l'URL. Ainsi, l'exemple précédent doit être complété automatiquement avec les paramètres "fontSize" et "fontColor".

Voici un exemple d'URL avec ces paramètres :

<https://cataas.com/cat/BYAFp9AIPCIDCKgf/says/chalut?fontSize=77&fontColor=orange>

Pour faire cela, dans **main.html** :



- Dans le formulaire, ajoutez une balise **fieldset** contenant
 - une balise **legend** qui permettra d'indiquer ce que signifie cette section du formulaire. Mettez-y le texte "Options de texte"
 - une balise **label** ayant le texte "Taille du texte" et l'attribut "for" avec la valeur "fontSize"
 - une balise **input** de type "range" (afin d'avoir un slider 😊) ayant l'identifiant "catTextSize" et les attributs suivants :

min="10" · max="100" · name="fontSize" · value="20"

où "min" indique la valeur entière minimale du slider et "max" en indique la valeur maximale, tandis que "value" indique la valeur initiale par défaut. Donc ici, la taille du texte sera de 20px par défaut. Pratique n'est-ce pas ?

- un champ d'insertion du texte dont voici le code (car il n'y a rien de nouveau) :

<label for="fontColor">Couleur du texte :</label>

<input type="text" id="catTextColor" name="fontColor" value="orange"/>

(vous y voyez que la couleur par défaut est "orange")

Maintenant, adaptez le code de **script.js** :



- créez une variable globale dédiée à contenir l'identifiant du chat 🐱. Pour cela, vous pouvez la créer avec une valeur **null**
- adaptez la fonction asynchrone afin de créer automatiquement l'URL avec les bons paramètres ajoutés s'il y en a
 - 💡 Pour cela, tous les choix seront simplement ajoutés dans l'objet du formulaire. C'est ce qui rend cette approche si pratique : pas besoin d'aller chercher chaque élément du DOM avec leur identifiant, on peut simplement récupérer l'ensemble des choix d'un formulaire en un objet 😊

Normalement, tout est bon. Votre interface a les fonctionnalités suivantes :

- chargement d'une image de chat aléatoire
- ajout du texte sur l'image de chat aléatoire ou sur celle en cours
- personnalisation de la taille et de la couleur du texte sur l'image de chat aléatoire ou celle en cours

Dans ce TD vous avez donc continué à utiliser les requêtes asynchrones avec *el famoso* triptyque **async await fetch()**, tout en indiquant bien vouloir recevoir un certain type de données. En plus de cela, vous avez vu une façon plus optimisée de regrouper des choix par le biais d'un formulaire qui ne sert ici qu'à regrouper les différentes sélections.

Je vous ai guidé dans ce TD, surtout vis-à-vis de l'usage de l'API. Mais sachez que pour toute API, il vous faudrait bien regarder la documentation pour savoir comment elles ont été conçues.

Enfin, voici un cadeau : vous pouvez obtenir une image animée (GIF) de chat aléatoire à l'adresse suivante : <https://cataas.com/cat/gif> (rechargez la page pour en voir d'autres)

C'est bon pour ce TD ! Youhou ! Vous en êtes venu à bout !



Liens utiles

- <https://developer.mozilla.org/fr/docs/Web/JavaScript>
- <https://www.w3schools.com/js/default.asp>
- <https://cataas.com/>