programmation web en js

projet Photobox : partie 1

Photobox : parcourez et visualisez des galeries de photos
Photobox est une application de parcours et de visualisation de photos, permettant d'afficher et de faire défiler des galeries de photos paginées. Un mode "lightbox", où chaque photo occupe l'ensemble de la fenêtre navigateur est également disponible et permet de parcourir les galeries page par page.

Préliminaires : Accéder aux données Photobox
Les données photobox, collections de photos, photos et leur description, catégories sont accessibles au travers de l'API photobox qui les retourne en format json.
Pour rappel, l'api est accessible sur ce point d'entrée :
https://webetu.iutnc.univ-lorraine.fr/www/canals5/photobox/
et la documentation complète disponible ici :
https://webetu.iutnc.univ-lorraine.fr/www/canals5/photobox/doc/

Rappel : utiliser le VPN, ajouter l'option `{ credentials : 'include' }` dans les fetch.

Exercice 1 : afficher 1 galerie de photos
L'objectif est d'afficher une liste de photos sous forme de vignettes. La liste de photos est obtenue grâce à une requête fetch auprès de l'api photobox.
La galerie est construite en vous basant sur le markup html et le code css fourni dans le squelette. Vous pouvez le modifier si vous le souhaitez.
Procédez par étapes :

1. Examiner la structure des données retournées par l'api lorsque l'on accède à une collection de photos avec une requête sur l'uri https://webetu.iutnc.univ-lorraine.fr/www/canals5/photobox/photos,
2. réutilisez et complétez si nécessaire le module `photoloader` du TD xhr pour gérer les requêtes vers l'api. En particulier, vous utiliserez la fonction loadResource() (td5, exercice 3) pour charger la liste de photos initiale. La méthode retourne une promesse permettant d'accéder aux données renvoyées par l'api.
3. créez un module nommé `gallery` pour le chargement et la pagination des galeries. Ce module exporte une fonction `load()` qui charge la liste de photos (en utilisant le module photoloader), stocke les données et retourne la galerie.
4. créez le module `gallery_ui` chargé de l'affichage d'une galerie. Le module exporte la fonction `display_galerie()` qui reçoit une galerie en paramètre et l'affiche en construisant le markup html correspondant, puis en l'insérant dans le DOM. Basez-vous sur le markup html fourni dans le squelette. Remarquez que la valeur de l'attribut `data-uri` est l'uri de la photo correspondant à la vignette.
5. Dans le module principal de l'application, associer l'événement "click" sur le bouton de chargement d'une galerie à l'action correspondante : charger puis afficher la galerie.

Exercice 2 : lightbox : affichage d'une photo
Lorsque l'utilisateur clique sur une photo de la galerie, on souhaite afficher cette photo en

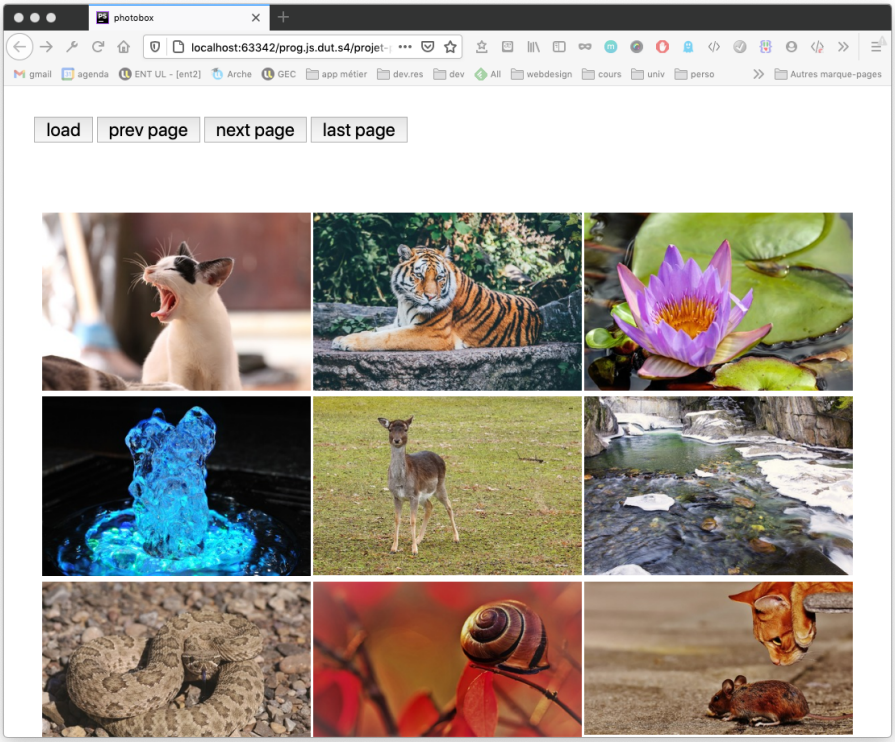
format original dans une "lightbox", c'est à dire dans un cadre venant recouvrir la galerie (voir exemple en annexe). La galerie doit rester présente, pour pouvoir réapparaitre lorsque l'utilisateur quitte la lightbox. Pour réaliser cette fonctionnalité, vous pouvez utiliser le markup html et le code css fourni dans le squelette qu'il faudra compléter.
Pour l'instant, on affiche uniquement le titre et l'image originale. Ces données (titre et l'url de la photo originale) doivent être récupérées grâce à une requête auprès de l'api. L'uri à utiliser est la valeur de l'attribut `data-uri` de la vignette qui a été sélectionnée.

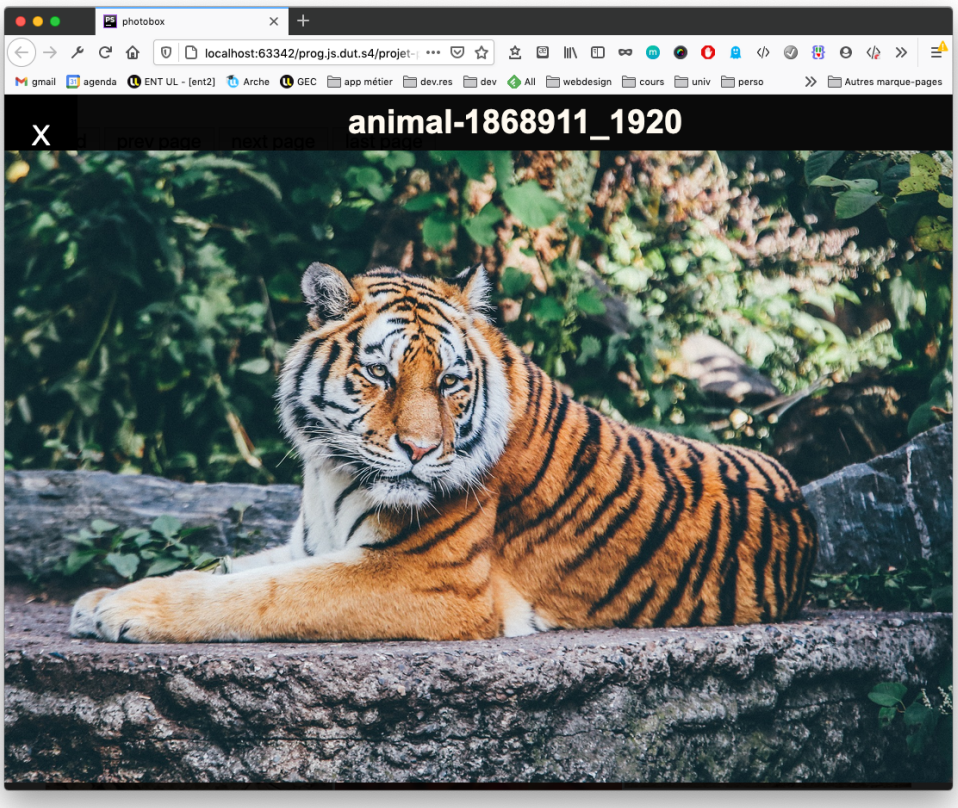
1. Créez un module `lightbox` chargé de gérer le contenu de la lightbox. Il esporte une fonction load(node) qui reçoit node (dom) correspondant à l'image cliquée par l'utilisateur, et charge les données de l'image en utilisant la valeur de l'attribut data-uri. Le chargement se fait en utilisant la fonction `loadResource( uri )` du module `photoloader` (td5).
2. Créez le module `lightbox_ui` chargé de l'affichage d'une lightbox. Le module exporte la fonction `display_lightbox( data )` qui reçoit les données décrivant une image et affiche la lightbox dans le conteneur prévu à cet effet (`#lightbox_container`) :
   - compléter le code css pour améliorer l'affichage du titre et du bouton pour quitter la lightbox (X),
   - prévoir 2 méthodes `show`() et `hide`() pour cacher/visualiser la lightbox,
   - programmer la fonction `display_lightbox` qui crée et insère le dom nécessaire puis rend visible la lightbox,
   - compléter avec la fonctionnalité permettant de fermer la lightbox par un click sur le bouton `#lightbox_close` ; on se contente de masquer la ligthbox.
3. compléter la méthode d'affichage d'une galerie pour qu'un click sur une image de la galerie déclenche l'affichage de la photo dans la lightbox.
   indication : la méthode `.querySelectorAll( ... )` retourne un tableau qui peut être traité avec `.forEach( ... )`

Exercice 3 : naviguer dans les galeries
On souhaite compléter la navigation dans les galerie de photos en ajoutant la possibilité de parcourir les pages de photos, en cliquant sur les boutons prévus à cet effet.

1. examiner les données retournées par l'api pour la liste de photos et identifier comment, à partir de ces données, on peut obtenir la page suivante et la page précédente.
2. compléter le module `gallery` pour y ajouter les méthodes permettant de charger la page suivante/précédente de la galerie. Pour cela, lors du chargement d'une galerie, on devra stocker les informations permettant la navigation.
   1. compléter la fonction load() déjà présente pour qu'elle stocke ces informations,
   2. programmer et exporter les méthodes next() et prev() qui chargent les données de la page suivante/précédente,
3. ajouter les listener pour les boutons "next" et "prev" dans la zone de navigation de votre interface html. Ils chargent puis affichent la galerie suivante/précédente.
4. sur le même principe, ajouter les bouton 'first" et "last" pour accéder à la première/dernière page.

Exercice 4 : affichage des informations détaillées sur une photo dans la lightbox
Compléter l'affichage des informations décrivant une image lorsqu'elle est affichée dans la lightbox. On affiche ces informations dans une zone dédiée placée sous la photo. Il faudra créer le markup html/css nécessaire.


photobox
localhost:63342/prog.js.dut.s4/projet-
gmail
agenda
ENT UL - [ent2]
Arche
GEC
app métier
dev.res
dev
All
webdesign
cours
univ
perso
Autres marque-pages
load
prev page
next page
last page


photobox
localhost:63342/prog.js.dut.s4/projet-
gmail
agenda
ENT UL - [ent2]
Arche
GEC
app métier
dev.res
dev
All
webdesign
cours
univ
perso
Autres marque-pages
X
animal-1868911_1920