**Réalisation d’une Picture box (boite à images)**

|  |  |
| --- | --- |
|  |  |

1. **Le projet**

La réalisation de ce projet s’appuie sur un cahier des charges et plusieurs vues fonctionnelles de type Wireframe.

L’interface doit être décomposé en 3 parties :

* Un accès aux différentes catégories
* Les vignettes regroupant les miniatures de chaque catégorie
* Un bloc affichant les détails de chaque vignette (photo, titre, commentaires)

Dans ce projet j’aborde les méthodes que j’ai mise en place dans le cadre d’une application single page, le code html étant généré dynamiquement.

Pour démarrer mon projet j’utilise VsCode un éditeur de code multiplateforme, open source et gratuit, supportant une dizaine de langages.

1. **Ecrire du code**
   1. **Html**

Je commence par créer un fichier HTML en m’assurant qu’il respectera les recommandations du w3C, organisme de standardisation chargé de promouvoir la compatibilité des technologies du World Wide Web.

Je décide d’utiliser le principe des grilles de Bootstrap pour contenir mes blocs éléments et obtenir une page responsive et adaptable.

Au début du fichier HTML j’indique le Doctype instruction qui décrit le type de document.

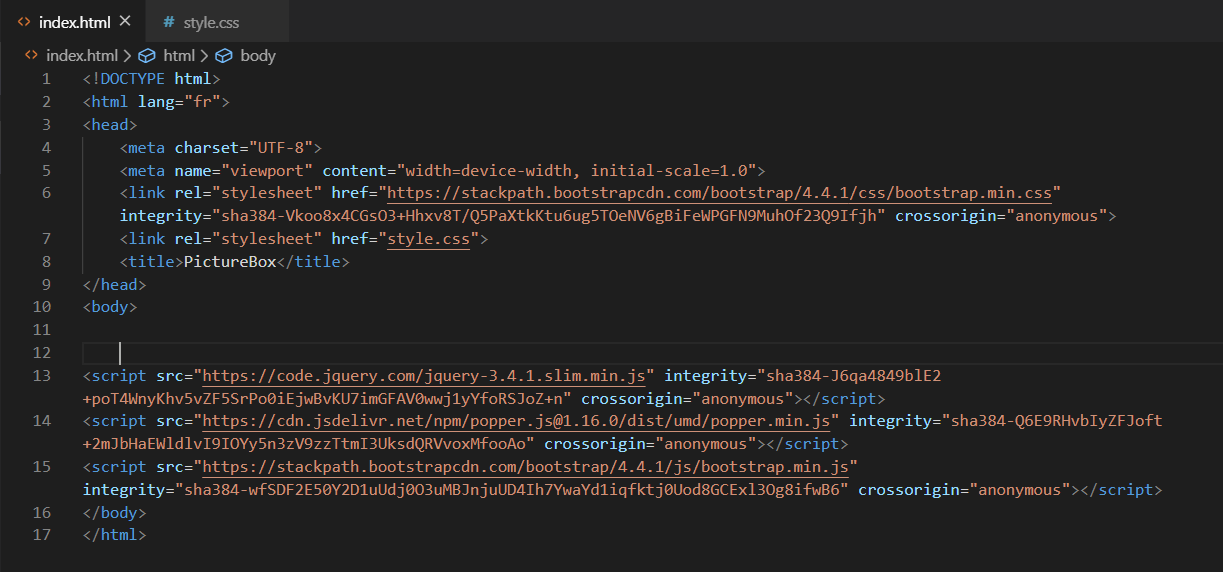
La balise HTML qui contient les balises enfants est divisée en deux parties :

L’entête : <head> partie invisible de la page.

* Je précise le format de codage des caractères utilisées Utf-8
* Un lien vers le CDN de Bootstrap un Framework CSS
* Le lien vers ma feuille de style
* Le titre de ma page.

Le corps de page : <body> partie visible de la page.

* J’ajoute les scripts nécessaires pour certaines fonctionnalités de Bootstrap.



Je crée un ensemble de balises sémantiques imbriquées contenant chacune des classes Bootstrap, un conteneur principal (container), une ligne (row), de colonnes (col) afin d’agencer le contenu de ma page.

La ligne contient 4 colonnes que je définis de type Flex :

* Ma première colonne est une balise sémantique <header> pour contenir le titre principal et un sous-titre
* La seconde regroupe l’ensemble des boutons permettant la navigation entre les différentes pages

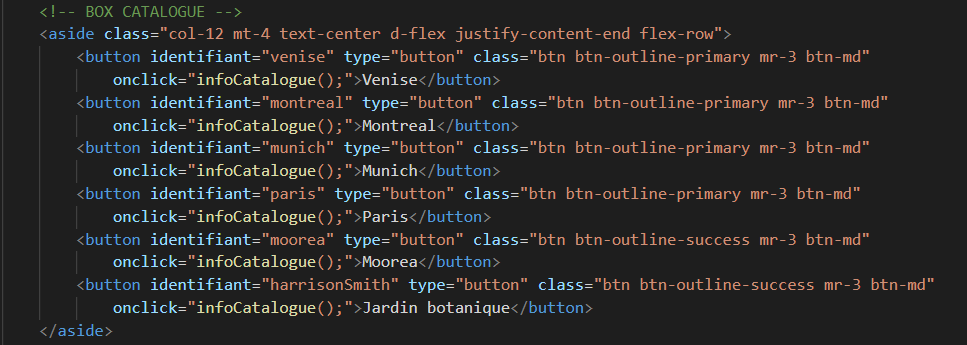
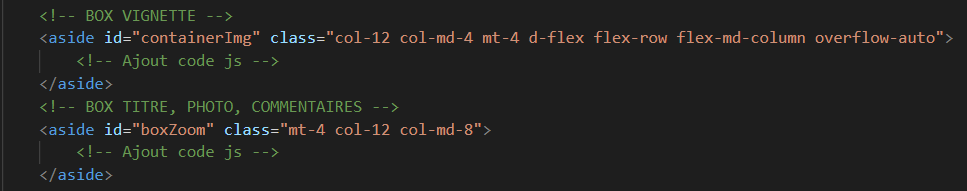
A chaque bouton j’ajoute un attribut « identifiant » avec pour valeur le nom de la ville

Un évènement « onclick » pointant sur une fonction de mon code JavaScript.

* La troisième dispense les photos sous forme de vignettes, avec une classe « overflow ».

Le contenu est généré à partir d’un fichier JavaScript.

* La dernière est un bloc dans lequel apparaît le titre, la photo et les commentaires associés.

**Code CSS**

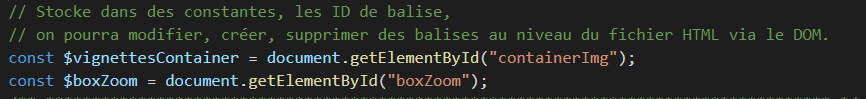
Un fichier CSS pour définir quelques règles :

* Le type de police,
* La taille des vignettes
* La taille maximum de la photo
* Et une animation au survol des vignettes

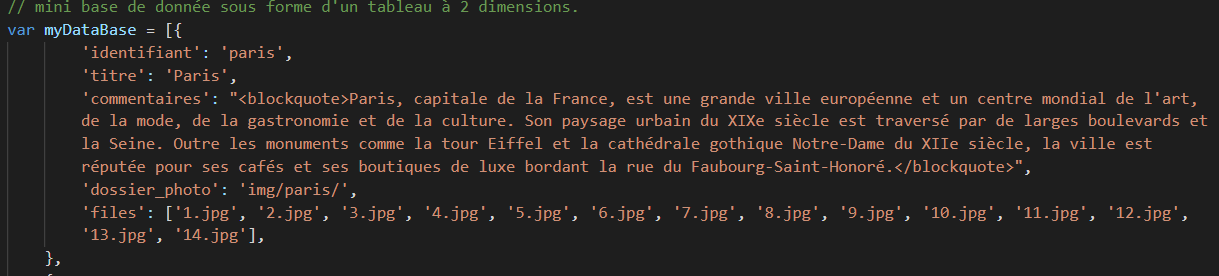
**Code JAVASCRIPT**

J’ajoute un fichier JavaScript afin de manipuler le DOM interface de programmation qui permet de manipuler les éléments d’un document HTML.

Je définis 2 constantes, pour chaque bloc parent : vignettes et détail.



Pour stocker toutes les informations : nom de ville, titre, commentaires et photos je crée un tableau d’objet avec des clefs identifiables au travers de mon code.



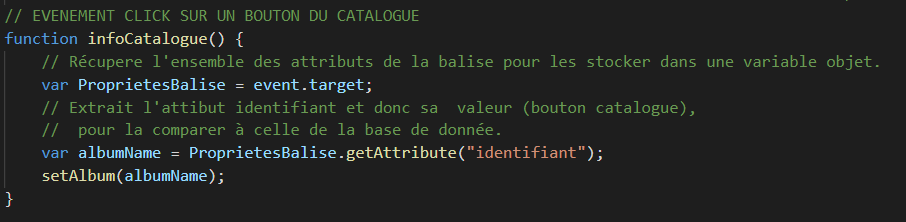
Au chargement de la page web, JavaScript exécute une instruction pointant sur une fonction, celle-ci reçoit en paramètre le nom de la ville et affiche les données associées.

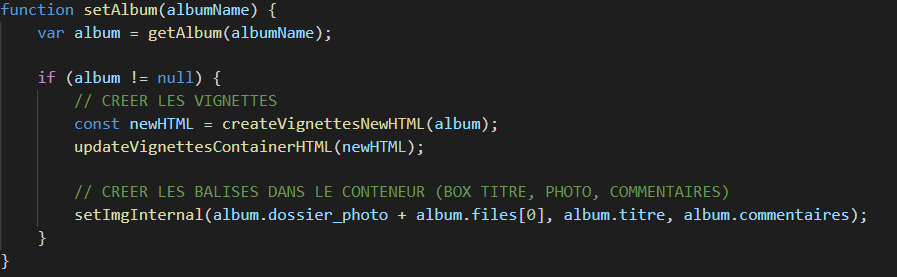
**Affichage d’une nouvelle page lors d’un clic sur le bouton ville**

Lors du clic sur le bouton ville la propriété **on click** appelle la fonction « info Catalogue ».



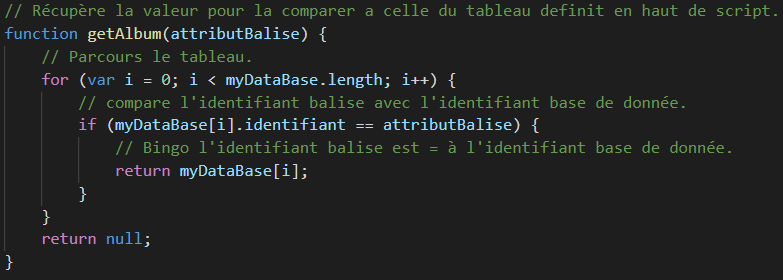
« Info Catalogue » écoute les évènements survenus sur le document, récupère l’élément puis j’extrais la valeur de l’attribut « identifiant » que j’envoie en paramètre vers la fonction « setAlbum ».





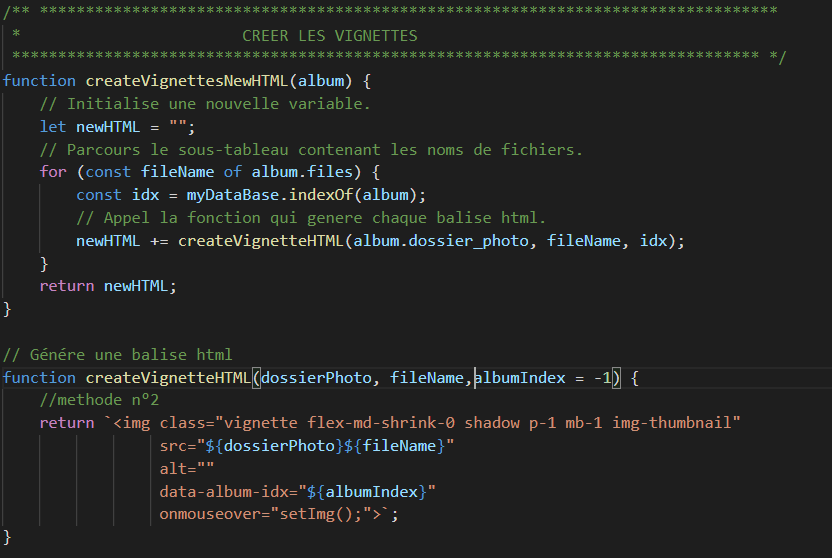
« setAlbum » appelle 4 fonctions :

* La fonction « getAlbum » ci-dessous va parcourir le tableau et comparer le nom de la ville passée en paramètre avec les clefs stocké dans le tableau, elle retourne un objet contenant les données associées à la recherche ou nulle si les valeurs ne correspondent pas.

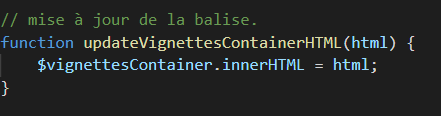


Maintenant que j’ai récupéré l’ensemble des informations appartenant à la ville je dois générer le code html dans leur bloc respectif.

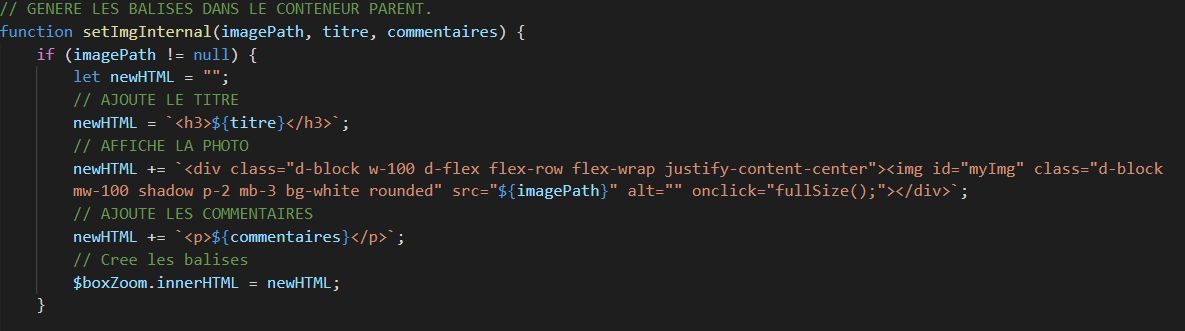
* J’actualise l’élément parent pour l’affichage des vignettes avec la seconde fonction « createVignettesNewHTML » qui parcours chaque index du tableau images, appelle la sous fonction « createVignetteHTML » pour générer le code HTML.



* Il ne me reste plus qu’à l’ajouter l’intégralité des vignettes le conteneur du document HTML.



* Pour afficher le reste des informations, le titre, l’image et les commentaires je procède de la même manière.



L’interface est fonctionnelle, j’ai ajouté la possibilité d’afficher l’image en pleine écran dans un modal.

**2. Précisez les moyens utilisés :**

Dans cette démonstration j’ai utilisé un éditeur de texte.

Effectué des recherches sur différents site web : w3schoom.com, MDN web docs de mozilla.

**3. Avec qui avez-vous travaillé ?**

Cet exercice était proposé dans le cadre de la mise en pratique de technologies Front-end web, je l’ai réalisé seul.