Projet Nesti 1
Documentation

I.a index

Pour commencer par le commencement. Tout d'abord il y a la création de l'HTML suivit des icône de fenêtre pour tout les formats d'appareils.

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Nesti</title>
    <link
      rel="shortcut icon"
      href="./images/icones/favicon.ico"
      type="image/x-icon"
    />
    <link rel="icon" href="./images/icones/favicon.png" type="image/png" />
    <link
      rel="icon"
      sizes="32x32"
      href="./images/icones/favicon-32.png"
      type="image/png"
    />
    <link
      rel="icon"
      sizes="64x64"
      href="./images/icones/favicon-64.png"
      type="image/png"
    />
```

Suivit des différents import lier aux fonctionnements :

```
<meta name="msapplication-TileColor" content="#FFFFFF" />

    <link rel="stylesheet" href="./styles/css-reset.css" />
```

Importation du plugin tailwinds

```
<link
      href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css"
      rel="stylesheet"
    />
```

Importation des fronts

```
    <link
      href="https://fonts.googleapis.com/css?
family=Raleway:100,100i,200,200i,300,300i,400,400i,500,500i,600,600i,700,700i,800,8
00i,900,900i"
      rel="stylesheet"
    />
    <link
      href="https://fonts.googleapis.com/css?family=Lora:400,400i,700,700i"
      rel="stylesheet"
```

I.b Le main

Le main est composé de trois éléments principaux:

- La modal

```
      <div id="myModal" class="modal">
        <div class="borderModal rounded">
          <div class="modal-content">
```

- La grille de recette

```
<div
            class="repice grid grid-cols-4 md:grid-cols-5 gap-1 md:gap-4 mt-0 md:m
t-20 mb-10 md:mb-0 place-self-center"
          ></div>
```

- Le corps pour les cartes.

```
<div class="grid lg:grid-cols-5 grid-cols-3 justify-center">
        <div class="flex flex-col lg:col-span-3 col-span-3 mt-2">
          <h2>Choisissez vos ingrédients</h2>
          <div class="flex flex-col justify-center">
```

Puis le « footer » et enfin les scripts.
Concernant le fonctionnement l'ensemble est principalement géré par les scripts javascript.

II. Le recipe.js et le creatObjets.js

Le fichier « recipe.js » à comme principale fonction de « stocker » les informations. Il est construit sous la forme :

```
var repices = [
  {
      "id":
      "name":
       "number":
      "time":
      "picture":
      "ingredients": [ , ],
      "lists": {
                    "Principale": [,]
            },
      "preparations": {
            "principale": [ , ],
      "Conseil":
            }
  }, Element 2, ect …]
```

Ce format contient l'ensemble des valeurs qui peuvent être appeler dans les fonctions afin de faciliter le code.

Dans le « CreateObjects.js » les informations « repices » sont extraites puis stockées sous forme de tableau

```
var arrayRepices = [];
repices.forEach((repice) => {
  arrayRepices.push(new Recipe(repice));
```

puis ont crée la liste des ingrédients

```
var listIngredients = [] ;
```

et ont fait de même.

```
var arrayCards = [];
listIngredients.forEach((ingredient) => {
  arrayCards.push(new Card(ingredient));
});
```

III. ListObjects.js

Dans ce fichier les objet Card et recipe sont crée affin d'attribuer les « informations » de chaque categorie du Json.

```
class Recipe {
  constructor(repice) {…}

class Card {
  constructor(ingredient) {...}
```

La fonction `validRecipe(ingredients)` compare les ingrédients validée par l'utilisateur avec ceux disponible dans les recette. Par défaut, tant que la liste des ingrédients sélectionnée par l'utilisateur 'keep' est vide elle retourne « False ».
`changeValid(value)` est utilisé pour modifier la valeur de `this.valid`
ce qui permettra dans la fonction `recipeDisplay()` d'aller chercher les recettes par l'intermédiaire de `searchRecipe()`.

`checkWriting(text)`
Cette fonction est présente afin d'éviter les confies dans le traitement des mots, elle supprime les accent et majuscule.

IV. function.js

Tout d'abord, une petite explication s'impose sur le fonctionnement. Lorsque l'utilisateur arrive sur la page, il peut cliquer sur Sweep ou Keep mais au niveau code l'ensemble des cartes sont déjà encodé et seul les 2 premières sont physiquement présente. Au moment du clique, la première page s'en va et fait sont effet pour revenir en fin...A ce moment la carte «3 » est physiquement crée (pour anticipée et ne pas crée un vide pour un nouveau clique).
`function possitionCard(pos)`

Rentrons dans le détails des fonctions...

```
function countCard() {
  let arrayCardValid = [];
  arrayCards.forEach((ingredient) => {
    if (ingredient.valid) {
      arrayCardValid.push(ingredient.name.toLowerCase());
    }
  });
  return arrayCardValid;
}
```

Cette fonction permet de conserver une liste (avec un texte épuré `toLowerCase` afin d'évité des erreurs) des cartes validé par l'utilisateur.

La fonction suivante utilise le retour de `countCard() >> arrayCardValid`

```
function searchRecipe() {
  var cards = countCard();
  for (var repice in arrayRepices) {
    arrayRepices[repice].validRecipe(cards);
```

Sont but est de valider les recettes `validRecipe` selon le choix des ingrédients utilisateur `arrayRepices`.

```
function randomize(array) {}
```
Cette fonction crée un orde aleatoire et est utilisé afin que les « Card » et les « recipes » ne soit pas toujours présent dans le même ordre initial.

```
function viewportSize() {}
```
Cette fonction contrôle la taille de la fenêtre utilisateurs et renvoie ses dimensions.

```
function possitionCard(pos) {}
```
Cette fonction conditionne les positions, image et taille de texte dans la 'grille' concernant les « Cards ». Le « Style » par défaut étant géré pour des petit écrans l'ensemble est adapté plus pour des écrans « grande taille ». La gestion des images pour le 'responsif' est également géré dans cette partie afin que les images ne prennent pas trop de place.

```
function moveCard(way, move)
```
Cette fonction à pour rôle de crée un effet de mouvement de rotation de lors de la translations de la première carte.

```
function recipeDisplay()
```
Fonction qui génère la partie HTML et gère le conditionnement de l'ouverture de la modal.
Elle fait appel à `searchRecipe` afin d'extraire les informations nécessaires pour la constitutions de la grille « recipe ».

```
function openModal(id)
```
La fonction construit la modal en HTML et gère la récupération des donnée Json. Les différentes boucles permettent de parcourir les listes et de partitionnée les informations.
```
      if (repiceModal.valid)
```
Cette partie de la fonction crée toute la partie haute de la modal (avant image). Pour la suite de la fonction il faut récupérer les informations contenues dans le Json. On parcours chacune des listes par des boucles « for » pour les entêtes et on parcours de nouveau pour intégrer le texte « informatif ». Puis on injecte dans le `modal.innerHTML = div;`

```
 //Click condition on the modal
    var close = document.querySelector(".close");
    close.onclick = function () {
      modal.style.display = "none";
    };
    window.onclick = function (event) {
      if (event.target == modal) {
        modal.style.display = "none";
      }
    };
    var close = document.querySelector(".modal");
    modal.onclick = function (event) {
      modal.style.display = "none";
    };
```
Ces dernière fonction gèrent la fermeture de la modal soit par le bouton « Close», soit en cliquant sur la fenêtre ou bien la modal.

V. Management.js

Ce fichier est le fichier principale qui regroupe les fonctionnalités et interactivités.

```
var posRotateCard = ["rotate-1", "rotate-2", "-rotate-1", "-rotate-2"];
let widthtWindow = viewportSize().width;
if (widthtWindow < 550) {
  var posRotate = posRotateCard;
  var move = 500;
} else {
  var posRotate = ["rotate-6", "rotate-12", "-rotate-6", "-rotate-12"];
  var move = 650;
}
```

Limite le déplacement de la Card et la rotation des autres lors de la translations en fonction de la taille de la fenêtre.

```
for (let i = 0; i < cards.length; i++) {
  cards[i].style.zIndex = i + 2;
}
cardsI.style.zIndex = 10;
```

Cette partie fait en sorte que les cartes Soit devant la grille et ranger dans l'orde.

```
arrayCards = randomize(arrayCards);
possitionCard(posCard);
```

Lance l'effet aléatoire des « Cards»

```
keep.addEventListener("click", function () {
  arrayCards[posCard].changeValid(true);
  posCard++;
  if (posCard == arrayCards.length) {
    posCard = 0;
  }
  moveCard("+", move);
  recipeDisplay();
});
```

Ajout de l'attribut au bouton Keep de lancer l'action de valider la recette de passer à la Card suivante et de relancer la création de la grille de recette à partir des données ingrédients conserver.

```
randomize(arrayRepices);
recipeDisplay();
```

Crée l'effet aléatoire des recettes et crée la grille initial.

VI. Les éléments extérieurs

`tailwindcss` : https://tailwindcss.com/

Tailwind est un framework qui nous a permit de crée plus facilement les Cards. Son fonctionnement est un 'responsif' inversé : il se base sur une window de petit taille. Il faut donc, non pas adapter le « style » pour un téléphone (exemple boostap) mas l'inverse.

VII. Les sources d'images

https://www.pexels.com
https://pixabay.com/fr/