Le code que vous avez fourni crée une interface utilisateur permettant d'effectuer des recherches sur Wikipédia via leur API, de traiter et d'afficher les résultats. Voici une explication détaillée et simple des différentes parties du code :

**JavaScript**

1. **Sélection des éléments HTML** :

javascript

const form = document.querySelector("form");

const input = document.querySelector("input");

const errorMsg = document.querySelector(".error-msg");

const resultsDisplay = document.querySelector(".results-display");

const loader = document.querySelector(".loader");

* + form : Sélectionne l'élément <form>.
  + input : Sélectionne l'élément <input>.
  + errorMsg : Sélectionne l'élément avec la classe error-msg.
  + resultsDisplay : Sélectionne l'élément avec la classe results-display.
  + loader : Sélectionne l'élément avec la classe loader.

1. **Écouteur d'événement pour le formulaire** :

javascript

form.addEventListener("submit", handleSubmit)

* + Ajoute un écouteur d'événement qui appelle la fonction handleSubmit lorsque le formulaire est soumis.

1. **Fonction handleSubmit** :

javascript

function handleSubmit(e) {

e.preventDefault();

if(input.value === "") {

errorMsg.textContent = "Wops, veuillez remplir l'input";

return;

} else {

errorMsg.textContent = "";

loader.style.display = "flex";

resultsDisplay.textContent = "";

wikiApiCall(input.value);

}

}

* + Empêche le rechargement de la page (e.preventDefault()).
  + Vérifie si le champ de recherche est vide, affiche un message d'erreur si c'est le cas.
  + Si le champ n'est pas vide, efface le message d'erreur, affiche le loader, et vide la zone d'affichage des résultats.
  + Appelle la fonction wikiApiCall avec la valeur de l'input.

1. **Fonction wikiApiCall** :

javascript

async function wikiApiCall(searchInput) {

try {

const response = await fetch(`https://en.wikipedia.org/w/api.php?action=query&list=search&format=json&origin=\*&srlimit=20&srsearch=${searchInput}`);

if (!response.ok) {

throw new Error(`${response.status}`);

}

const data = await response.json();

createCards(data.query.search);

} catch (error) {

errorMsg.textContent = `$(error)`;

loader.style.display = "none";

}

}

* + Effectue une requête à l'API Wikipédia avec la chaîne de recherche fournie (searchInput).
  + Si la réponse n'est pas correcte (code de statut différent de 200), lance une erreur.
  + Parse les données JSON reçues.
  + Appelle la fonction createCards avec les résultats de la recherche.
  + En cas d'erreur, affiche un message d'erreur et cache le loader.

1. **Fonction createCards** :

javascript

function createCards(data) {

if(!data.length) {

errorMsg.textContent = "Wopsy, aucun résultat";

loader.style.display = "none";

return;

}

data.forEach(el => {

const url = `https://en.wikipedia.org/?curid=${el.pageid}`;

const card = document.createElement("div");

card.className = "result-item";

card.innerHTML = `

<h3 class="result-title">

<a href=${url} target="\_blank">${el.title}</a>

</h3>

<a href=${url} class="result-link" target="\_blank">${url}</a>

<span class="result-snippet">${el.snippet}</span>

<br>

`;

resultsDisplay.appendChild(card);

});

loader.style.display = "none";

}

* + Vérifie si les données sont vides et affiche un message d'erreur si c'est le cas.
  + Pour chaque résultat de recherche, crée un élément div avec la classe result-item.
  + Ajoute un titre, un lien et un extrait pour chaque résultat.
  + Ajoute chaque carte à l'élément resultsDisplay.
  + Cache le loader.

**CSS**

Le code CSS définit le style de la page et des différents éléments. Voici quelques points clés :

1. **Réinitialisation des marges et des paddings** :

css

\*,

::before,

::after {

box-sizing: border-box;

margin: 0;

padding: 0;

}

* + Applique box-sizing: border-box à tous les éléments, ainsi qu'à leurs pseudo-éléments ::before et ::after.
  + Réinitialise les marges et les paddings pour tous les éléments.

1. **Style global du body** :

css

body {

min-height: 100vh;

background: #f1f1f1;

font-family: Arial, Helvetica, sans-serif;

padding: clamp(20px, 5vw, 50px) 20px 20px;

}

* + Définit une hauteur minimale de 100vh pour le body.
  + Applique une couleur de fond (#f1f1f1).
  + Définit une police de caractères.
  + Ajoute du padding avec une valeur adaptative (clamp).

1. **Style pour le logo et le titre** :

css

.logo {

width: clamp(100px, 25vw, 250px);

display: block;

margin: 50px auto 0;

}

h1 {

font-size: clamp(20px, 5vw, 50px);

color: #333;

text-align: center;

margin: 20px 0 10px;

font-weight: 100;

}

h1 span {

font-weight: 800;

}

* + Ajuste la taille du logo et du titre selon la taille de l'écran.
  + Centre le titre et modifie son style.

1. **Style pour le formulaire et l'input** :

css

form {

max-width: 800px;

margin: 0 auto 30px;

}

input {

display: block;

width: 100%;

margin: 0 auto;

padding: 15px;

font-size: 18px;

border: none;

border-radius: 3px;

box-shadow: 0 5px 5px rgba(0,0,0,0.2);

}

* + Centre le formulaire et limite sa largeur maximale.
  + Style l'input pour qu'il occupe toute la largeur disponible avec un padding et une ombre portée.

1. **Messages d'erreur et affichage des résultats** :

css

.error-msg {

text-align: center;

color: #454545;

}

.results-display {

max-width: 600px;

margin: 70px auto;

}

* + Centre le message d'erreur et change sa couleur.
  + Centre l'affichage des résultats et limite sa largeur.

1. **Loader et animation** :

css

.loader {

display: flex;

justify-content: center;

}

.dot {

width: 15px;

height: 15px;

border-radius: 50%;

background: #111;

margin: 0 3px;

animation: load 0.3s infinite ease-out alternate;

}

.dot:nth-child(2) {

animation-delay: 0.1s;

}

.dot:nth-child(3) {

animation-delay: 0.2s;

}

@keyframes load {

to {

transform: translateY(10px);

}

}

* + Définit le style des points de chargement (loader) avec une animation de rebond.

1. **Style pour les éléments de résultats** :

css

.result-item {

margin-bottom: 20px;

}

.result-title {

font-size: 22px;

}

.result-snippet {

font-size: 15px;

color: #444;

}

.result-link {

color: #006621;

text-decoration: none;

display: block;

overflow-wrap: break-word;

}

* + Définit le style des éléments de résultats (cartes) avec des marges et des styles de texte spécifiques.

Ensemble, ces scripts et styles créent une interface utilisateur interactive pour rechercher des articles Wikipédia et afficher les résultats de manière esthétique.