EXERCICES SUR LES REQUÊTES API

Exercice 1 : Explorateur de régions, départements et communes

On utilisera l'API Geo fournie par api.gouv.fr pour explorer les différentes entités administratives de la France.

Objectif :

Créez une page web qui permet à l'utilisateur d'explorer les régions, départements et communes de la France. L'utilisateur doit pouvoir :

- Sélectionner une région depuis un menu déroulant.
- Après avoir sélectionné une région, affichez un autre menu déroulant avec la liste des départements de cette région.
- Après avoir sélectionné un département, affichez un dernier menu déroulant avec la liste des communes de ce département.

Testez l'application :

- Ouvrez la page HTML dans un navigateur.
- Sélectionnez une région, puis observez comment le menu déroulant des départements est mis à jour.
- Sélectionnez un département, puis observez comment le menu déroulant des communes est mis à jour.

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Explorateur Géo</title>
</head>
<body>

<select id="regions">
    <option value="">Sélectionnez une région</option>
    <!-- Les régions seront ajoutées ici -->
</select>

<select id="departments" disabled>
    <option value="">Sélectionnez un département</option>
    <!-- Les départements seront ajoutés ici -->
</select>

<select id="communes" disabled>
    <option value="">Sélectionnez une commune</option>
```

```
    <!-- Les communes seront ajoutées ici -->
</select>

<script src="script.js"></script>
</body>
</html>

function populateDropdown(data, dropdownId) {
    const dropdown = document.getElementById(dropdownId);
    dropdown.innerHTML = `<option
value="">Sélectionnez</option>`;
    data.forEach(item => {
        const option = document.createElement('option');
        option.value = item.code;
        option.textContent = item.nom;
        dropdown.appendChild(option);
    });
    dropdown.disabled = false;
}

fetch('https://geo.api.gouv.fr/regions')
    .then(response => response.json())
    .then(data => {
        populateDropdown(data, 'regions');
    });

document.getElementById('regions').addEventListener('change',
function() {
    const regionCode = this.value;
    if (!regionCode) return;


fetch(`https://geo.api.gouv.fr/regions/${regionCode}/departeme
nts`)
        .then(response => response.json())
        .then(data => {
            populateDropdown(data, 'departments');
        });
});

document.getElementById('departments').addEventListener('chang
e', function() {
    const deptCode = this.value;
    if (!deptCode) return;


fetch(`https://geo.api.gouv.fr/departements/${deptCode}/commun
es`)
        .then(response => response.json())
        .then(data => {
```

```
            populateDropdown(data, 'communes');
        });
});
```

Exercice 2 : Trouveur de fabricant d'adresse MAC

Objectif :

Créez une page web avec un formulaire permettant à l'utilisateur de saisir une adresse MAC. Lorsque l'utilisateur soumet le formulaire, utilisez l'API macvendors.com pour obtenir et afficher le nom du fabricant du périphérique correspondant à cette adresse MAC.

Étapes :

1. Créez le formulaire HTML.
2. Créez le JavaScript (script.js).

Testez l'application :

- Ouvrez la page HTML dans un navigateur.
- Saisissez une adresse MAC valide (par exemple, 00:1A:78:00:00:01).
- Cliquez sur "Trouver le fabricant" et observez le résultat affiché en dessous du formulaire.

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Trouveur de Fabricant MAC</title>
</head>
<body>

<form id="macForm">
    <label for="macAddress">Adresse MAC:</label>
    <input type="text" id="macAddress" name="macAddress"
required placeholder="00:00:00:00:00:00">
    <br><br>
    <input type="submit" value="Trouver le fabricant">
</form>

<div id="result"></div>

<script src="script.js"></script>
</body>
</html>
```

```
document.getElementById('macForm').addEventListener('submit',
function(e) {
    e.preventDefault();  // Empêche le rechargement de la page

    const macAddress =
document.getElementById('macAddress').value;

    fetch(`https://api.macvendors.com/${macAddress}`)
        .then(response => {
            if (!response.ok) {
                throw new Error("Adresse MAC non trouvée ou
format invalide.");
            }
            return response.text();
        })
        .then(data => {
            document.getElementById('result').textContent =
`Fabricant : ${data}`;
        })
        .catch(error => {
            document.getElementById('result').textContent =
`Erreur : ${error.message}`;
        });
});
```

Exercice 3 : Générateur de blagues

Objectif :

Créez une page web avec un bouton qui, lorsqu'il est cliqué, affiche une nouvelle blague à l'utilisateur en utilisant l'API jokeapi.dev.

Étapes :

1. Créez le squelette HTML :
2. Créez le JavaScript (script.js) :

Testez l'application :

- Ouvrez la page HTML dans un navigateur.
- Cliquez sur "Donne-moi une blague !" et observez une blague qui s'affiche en dessous du bouton.

En supplément :

- Vous pouvez également ajouter des options pour permettre à l'utilisateur de filtrer les blagues par catégorie (par exemple, programmer, général, etc.).
- Prenez en compte la gestion des erreurs. Si l'API est indisponible ou si une autre erreur survient, assurez-vous d'afficher un message d'erreur approprié à l'utilisateur.

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Générateur de Blagues</title>
</head>
<body>
<select id="jokeCategory">
    <option value="Any">Toutes les catégories</option>
    <option value="Programming">Programmation</option>
    <option value="Miscellaneous">Divers</option>
    <!-- Vous pouvez ajouter d'autres catégories ici -->
</select>

<button id="generateJoke">Donne-moi une blague !</button>
<p id="jokeText"></p>

<script src="script.js"></script>
</body>
</html>
document.getElementById('generateJoke').addEventListener('clic
k', function() {
    const category =
document.getElementById('jokeCategory').value;

    fetch(`https://v2.jokeapi.dev/joke/${category}`)
        .then(response => {
            if (!response.ok) {
                throw new Error("Erreur lors de la
récupération de la blague.");
            }
            return response.json();
        })
        .then(data => {
            if (data.type === "single") {

document.getElementById('jokeText').textContent = data.joke;
            } else if (data.type === "twopart") {

document.getElementById('jokeText').textContent =
`${data.setup} ... ${data.delivery}`;
            }
        })
        .catch(error => {
            document.getElementById('jokeText').textContent =
`Erreur : ${error.message}`;
        });
});
```

Exercice 4 : Création d'une API pour gérer des notes

Objectif :

Créez une API qui permet de gérer une collection de notes. L'API doit permettre de créer, lire, mettre à jour et supprimer des notes.

Pré-requis :

- Connaissance de base de JavaScript
- Node.js installé
- npm (Node Package Manager) installé

Étapes :

Initialisation du projet :

- *Créez un nouveau dossier pour votre projet.*

Installation des dépendances :

- *Installez Express et le body-parser (pour lire le corps des requêtes POST) :*

Création de l'API :

- *Créez un fichier server.js*

Testez votre API :

Utiliser curl pour tester les différentes routes de votre API.

- Créer une nouvelle note (POST):

```
curl -X POST -H "Content-Type: application/json" -d
'{"text":"Ma première note"}' http://localhost:3000/notes
```

Cette requête ajoute une nouvelle note avec le texte "Ma première note" à la liste des notes.

- Lire toutes les notes (GET):

```
curl http://localhost:3000/notes
```

Cette requête retourne toutes les notes disponibles dans la liste.

- Lire une note spécifique par son ID (GET): Prenons par exemple l'ID 1.

```
curl http://localhost:3000/notes/1
```

Cette requête retourne la note avec l'ID 1.

- Mettre à jour une note (PUT): Pour mettre à jour le texte de la note ayant l'ID 1.

```
curl -X PUT -H "Content-Type: application/json" -d
'{"text":"Ma note mise à jour"}' http://localhost:3000/notes/1
```

Cette requête modifie le texte de la note avec l'ID 1 pour le remplacer par "Ma note mise à jour".

- Supprimer une note (DELETE): Pour supprimer la note ayant l'ID 1.

```
curl -X DELETE http://localhost:3000/notes/1
```

Cette requête supprime la note avec l'ID 1 de la liste.

```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

// Middleware pour lire le JSON du corps des requêtes
app.use(bodyParser.json());

// Stockage simplifié en mémoire
let notes = [];
let currentId = 1;

// Routes

// GET: Liste des notes
app.get('/notes', (req, res) => {
    res.json(notes);
});

// POST: Ajouter une note
app.post('/notes', (req, res) => {
    const note = {
        id: currentId++,
        text: req.body.text
    };
    notes.push(note);
    res.json(note);
});

// GET: Lire une note spécifique par ID
app.get('/notes/:id', (req, res) => {
    const note = notes.find(n => n.id ===
parseInt(req.params.id));
    if (!note) return res.status(404).send('Note not found');
    res.json(note);
});

// PUT: Mettre à jour une note
app.put('/notes/:id', (req, res) => {
    const note = notes.find(n => n.id ===
parseInt(req.params.id));
    if (!note) return res.status(404).send('Note not found');

    note.text = req.body.text;
    res.json(note);
});
```

```
// DELETE: Supprimer une note
app.delete('/notes/:id', (req, res) => {
    const index = notes.findIndex(n => n.id ===
parseInt(req.params.id));
    if (index === -1) return res.status(404).send('Note not
found');

    notes.splice(index, 1);
    res.status(204).send();
});

const PORT = 3000;
app.listen(PORT, () => {
    console.log(`Server is running on
http://localhost:${PORT}`);
});
```