Introduction à fetch, à l'Interaction avec les APIs et à la Manipulation du JSON

Ce cours vise à enseigner l'utilisation de fetch pour interagir avec des APIs en utilisant les méthodes HTTP et manipuler les données JSON. Ces compétences sont essentielles pour l'échange de données entre applications web.

Introduction aux APIs et à fetch

Qu'est-ce qu'une API ?

Interface de Programmation d'Application permettant la communication entre applications.

Introduction à fetch

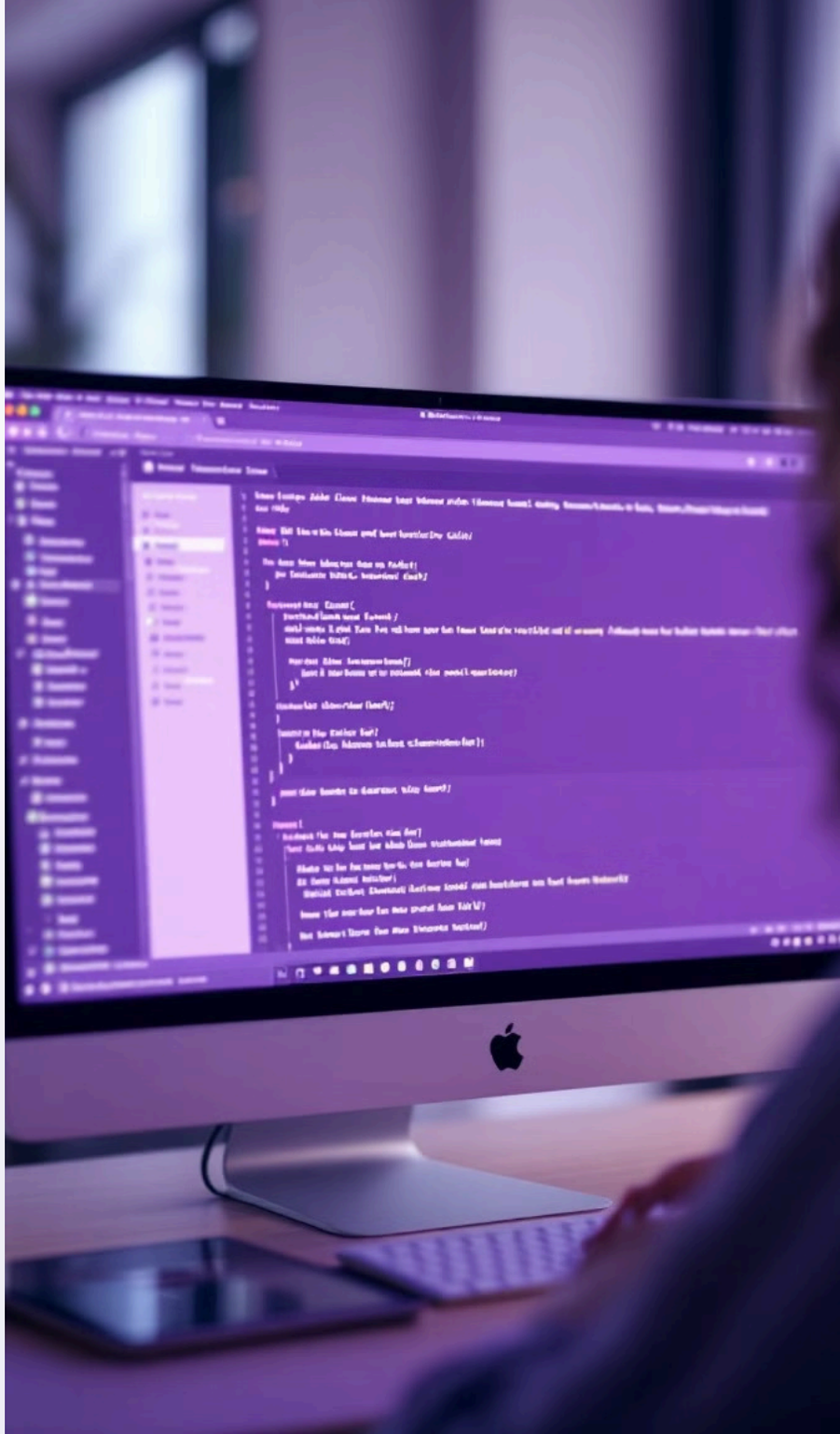
Méthode moderne pour effectuer des requêtes HTTP, remplaçant XMLHttpRequest.

Les méthodes HTTP

GET, POST, PUT, DELETE, OPTIONS pour interagir avec les ressources du serveur. (Il y en a d'autre mais plus rare)

Exemple de requête simple avec `fetch`

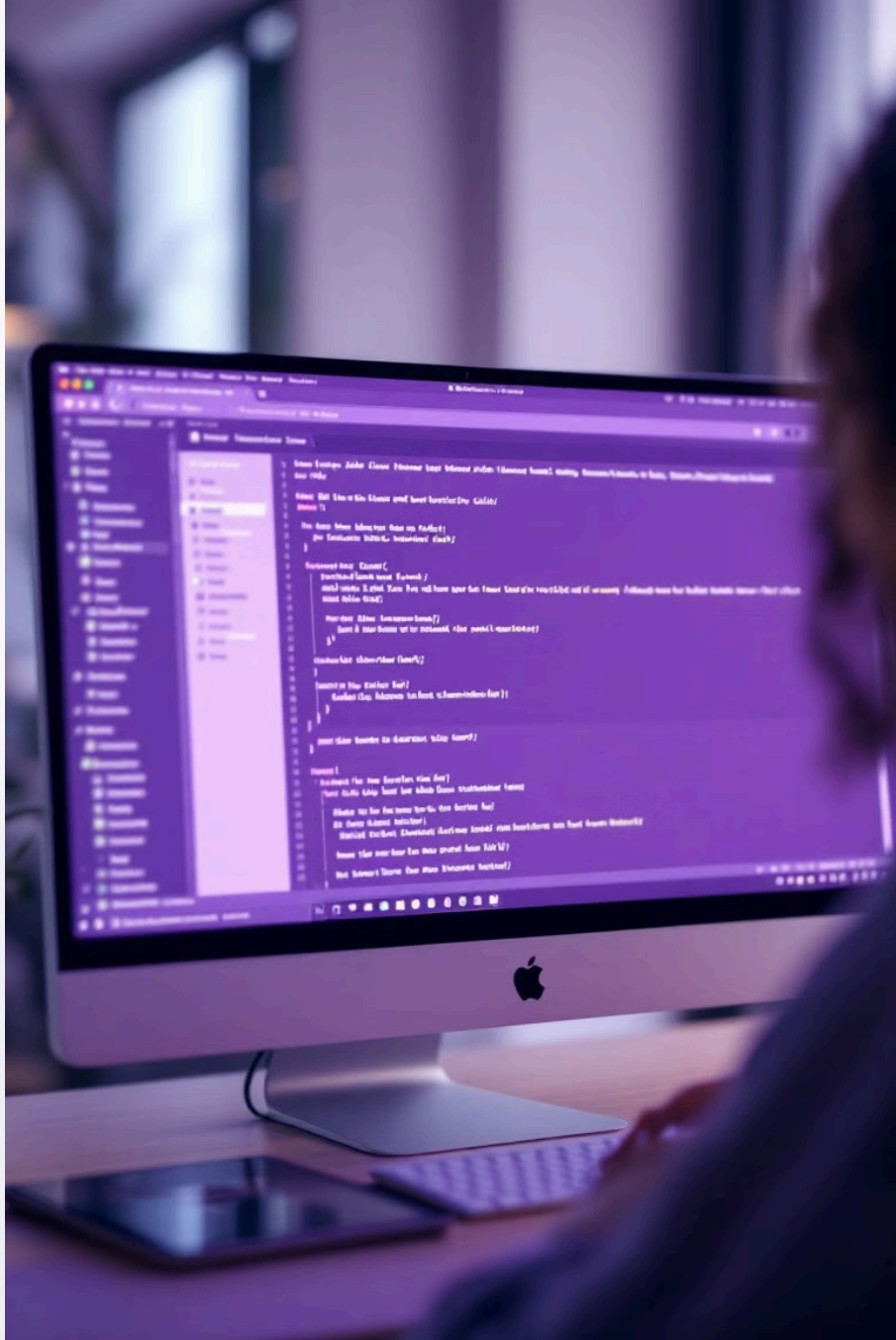
```
fetch('https://api.example.com/data')
    .then(response => response.json())
    .then(data => console.log(data))
    .catch(error => console.log('Erreur:', error));
```



Activité

Faire une requête GET vers une API publique

Utiliser cette api pour récupérer une liste d'articles:
https://jsonplaceholder.typicode.com/



Comprendre les Réponses et le JSON

Statuts des réponses HTTP

200 (Succès), 404 (Non trouvé), 500 (Erreur serveur). Vérifiez le statut avant de traiter les données.

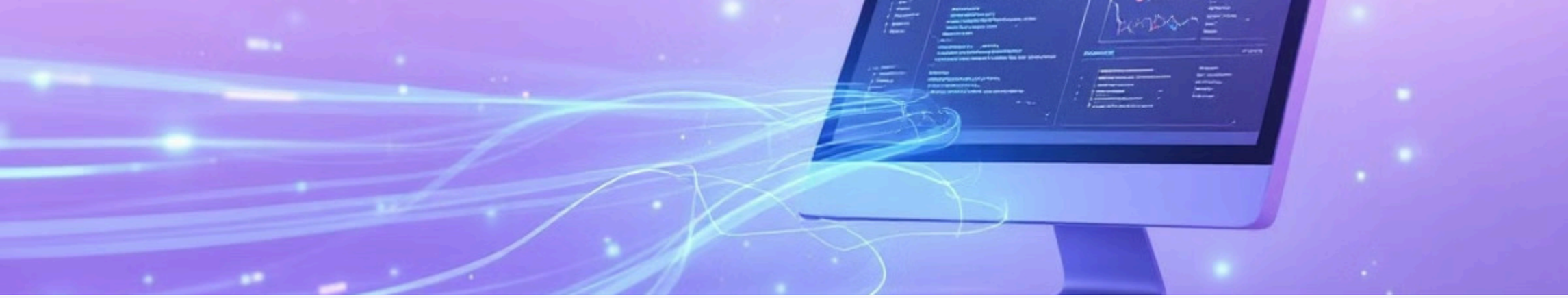
Comment vérifier le statut d'une réponse avant de traiter les données:

```
fetch('https://api.example.com/data')
    .then(response => {
        if (!response.ok) {
            throw new Error('Erreur réseau');
        }
        return response.json();
    })
    .then(data => console.log(data))
    .catch(error => console.log('Erreur:', error));
```

Manipulation des données JSON

Utilisez response.json() pour convertir la réponse en objet JavaScript. Accédez aux propriétés comme post.title.

```
fetch('https://jsonplaceholder.typicode.com/posts')
    .then(response => response.json())
    .then(data => {
        data.forEach(post => {
            console.log(post.title); // Accéder à chaque titre
        });
    })
    .catch(error => console.log('Erreur:', error));
```



Envoyer des Données avec POST

1

Utilisation de POST

Envoyez des données au serveur avec la méthode POST.

2

Conversion en JSON

Utilisez JSON.stringify() pour convertir l'objet JavaScript en JSON.

3

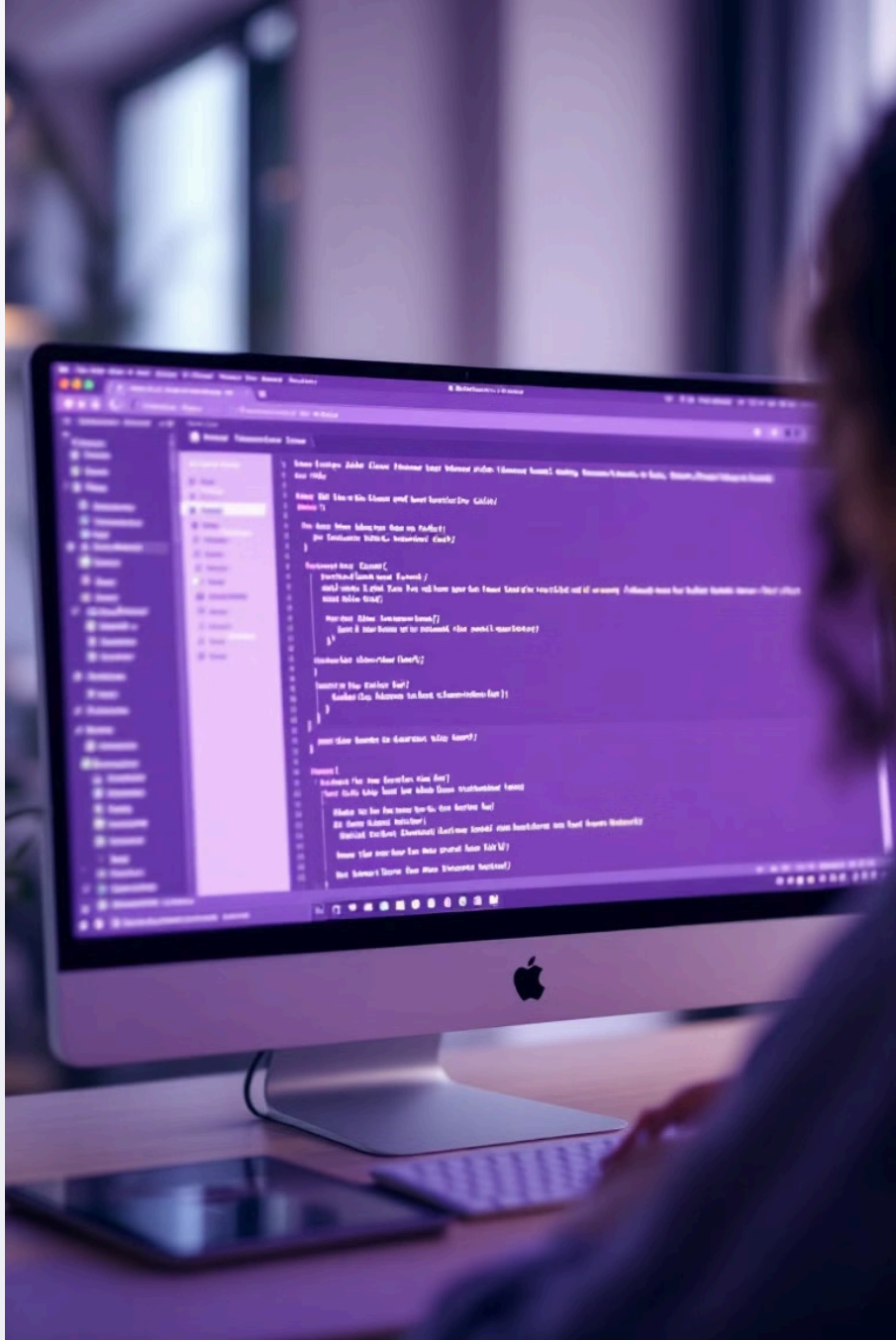
Envoi de la requête

Utilisez fetch avec la méthode POST et les en-têtes appropriés.

Activité

- [ ] **Créer un formulaire HTML pour ajouter un article**

  Envoyer les données via `POST` avec `fetch`.



Utiliser les Méthodes PUT et DELETE

Méthode PUT

Utilisée pour modifier une ressource existante sur le serveur.

```
fetch('https://jsonplaceholder.typicode.com/posts/1', {
    method: 'PUT',
    headers: {
        'Content-Type': 'application/json'
    },
    body: JSON.stringify({
        id: 1,
        title: 'Titre mis à jour',
        body: 'Contenu de l\'article mis à jour',
        userId: 1
    })
})
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.log('Erreur:', error));
```

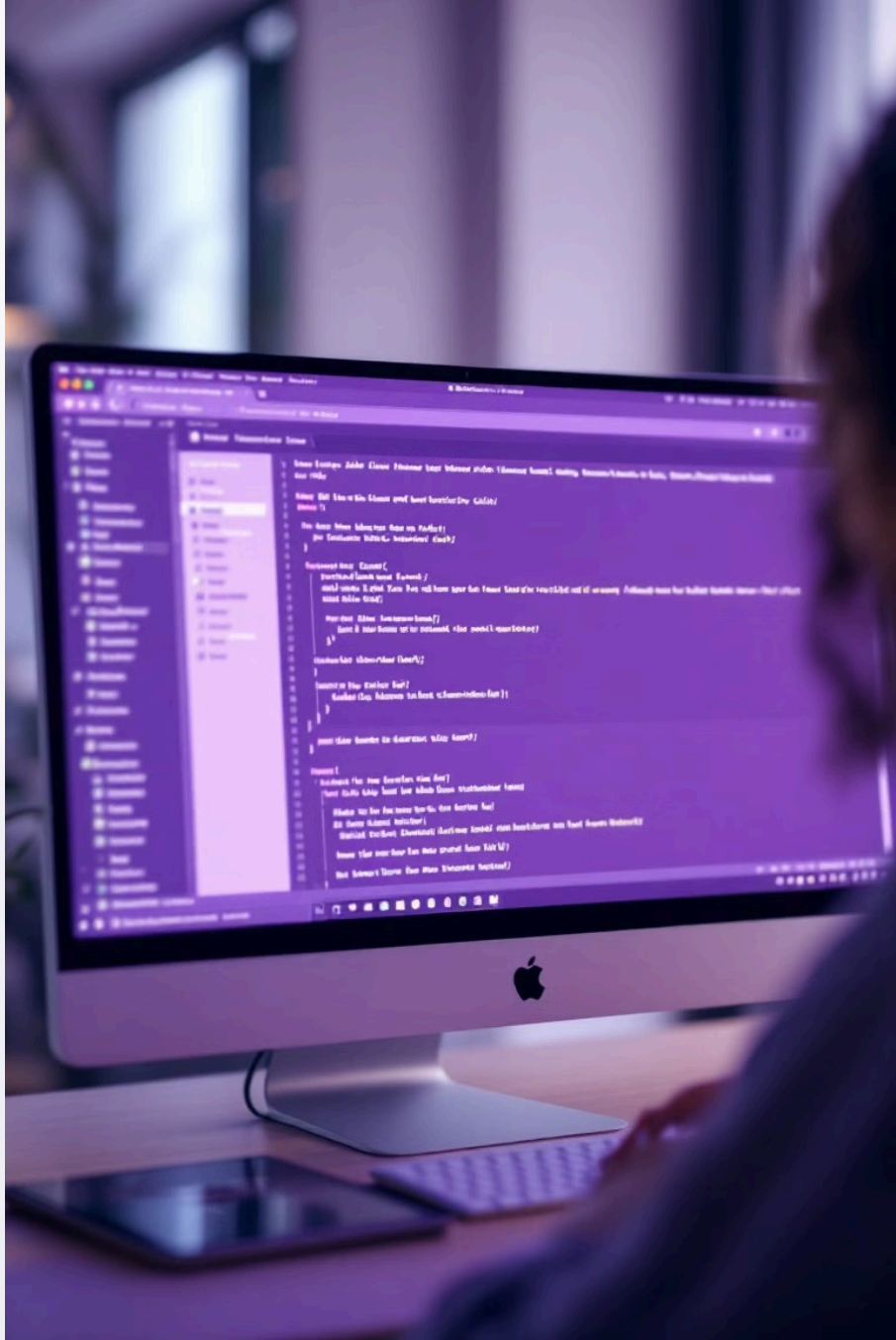
La méthode DELETE

Utilisation de DELETE pour supprimer une ressource du serveur.

```
fetch('https://jsonplaceholder.typicode.com/posts/1', {
    method: 'DELETE'
})
.then(response => {
    if (response.ok) {
        console.log('Article supprimé');
    }
})
.catch(error => console.log('Erreur:', error));
```

Activité

- [ ] Mettre à jour un article avec `PUT` et supprimer un article avec `DELETE`



Gérer les Erreurs et les Exceptions

1

Utilisation de catch()

Interceptez les erreurs réseau avec catch().

2

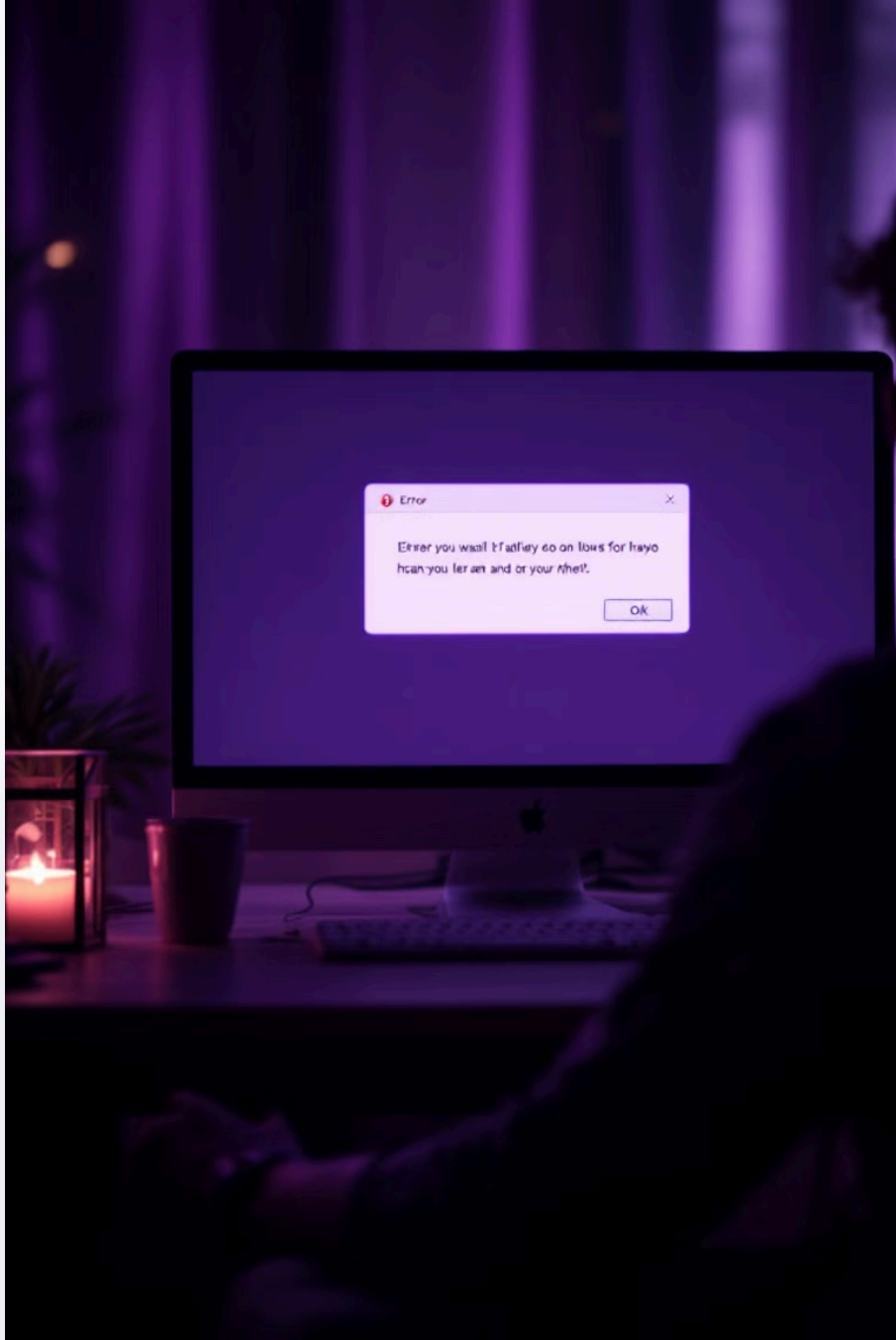
Gestion des erreurs HTTP

Gérez les statuts 404, 500, etc. avec des structures conditionnelles.

3

Messages d'erreur

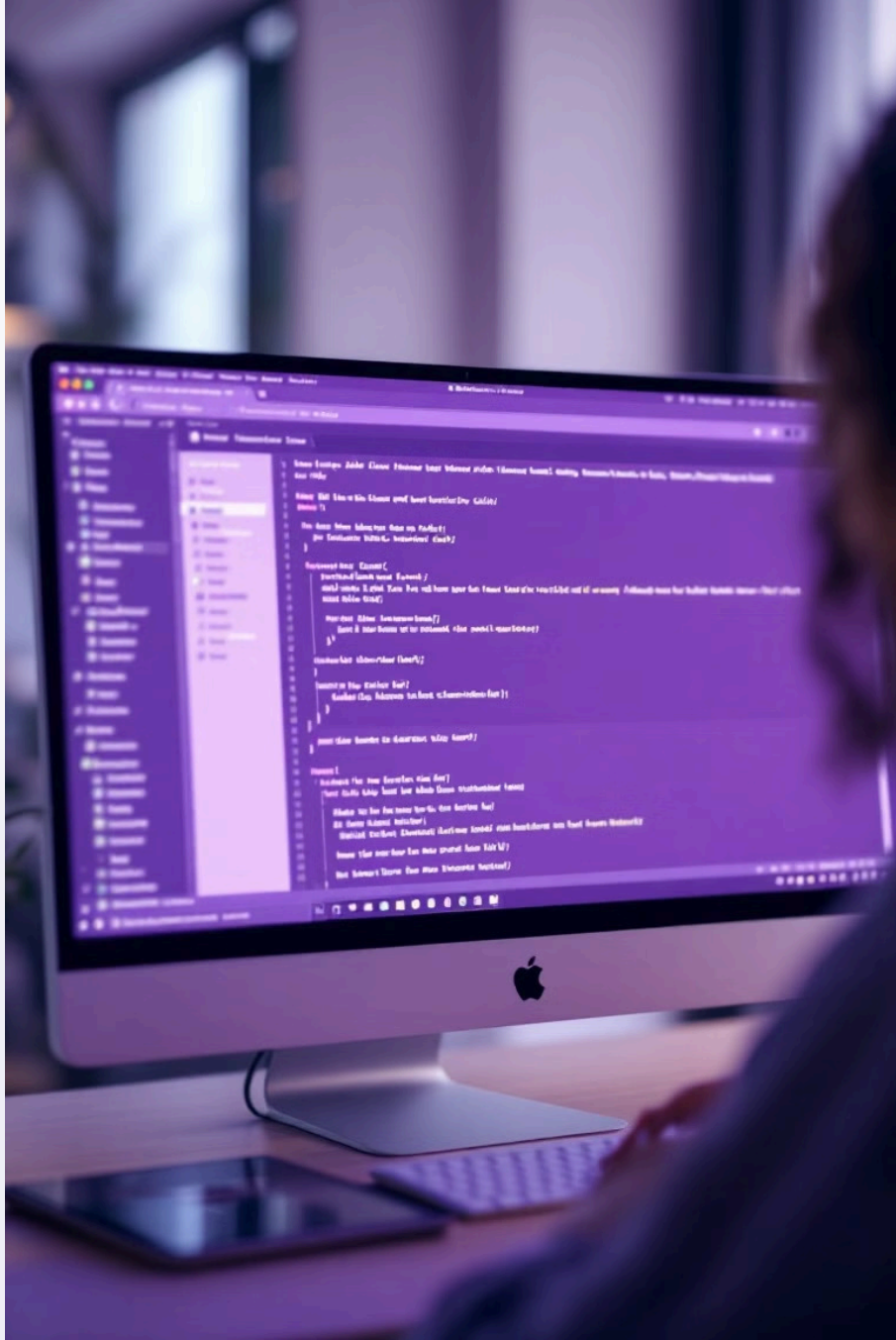
Affichez des messages d'erreur conviviaux pour l'utilisateur.



Activité

- [ ] **Faire une requête vers une API qui pourrait échouer**

  Afficher un message d'erreur à l'utilisateur en cas d’échec.



Sécuriser les Requêtes et Manipuler les En-têtes

En-têtes HTTP

Utilisez Content-Type et Authorization pour sécuriser les requêtes.

CORS

Comprenez et résolvez les problèmes de Cross-Origin Resource Sharing.

Authentification

Envoyez des tokens d'authentification dans les en-têtes pour les API protégées.

Authentification avec Authorization

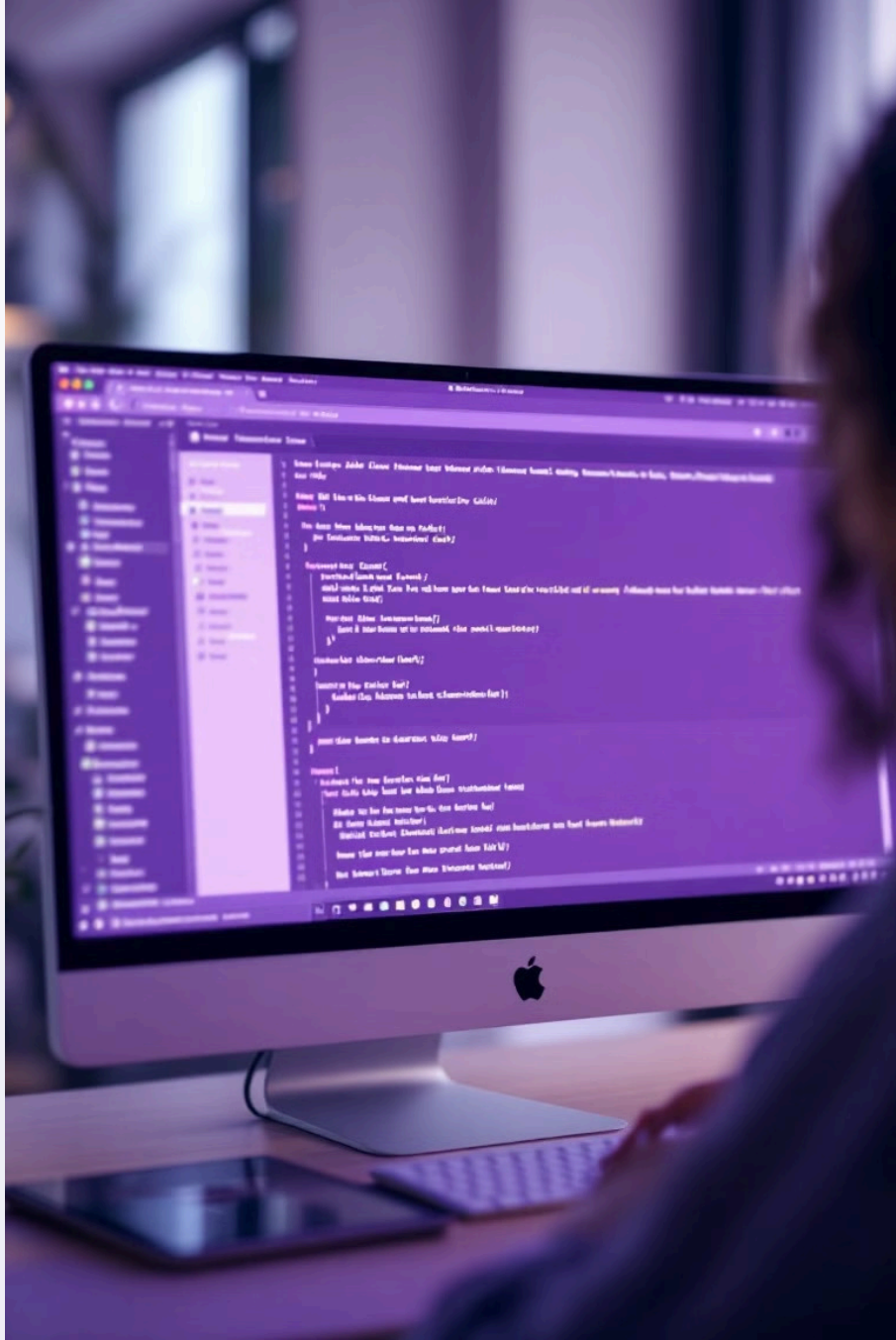
Exemple d'envoi d'un token d'authentification dans les en-têtes.

```
fetch('https://api.example.com/secure-data', {
    method: 'GET',
    headers: {
        'Authorization': 'Bearer <your-token>'
    }
})
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.log('Erreur:', error));
```

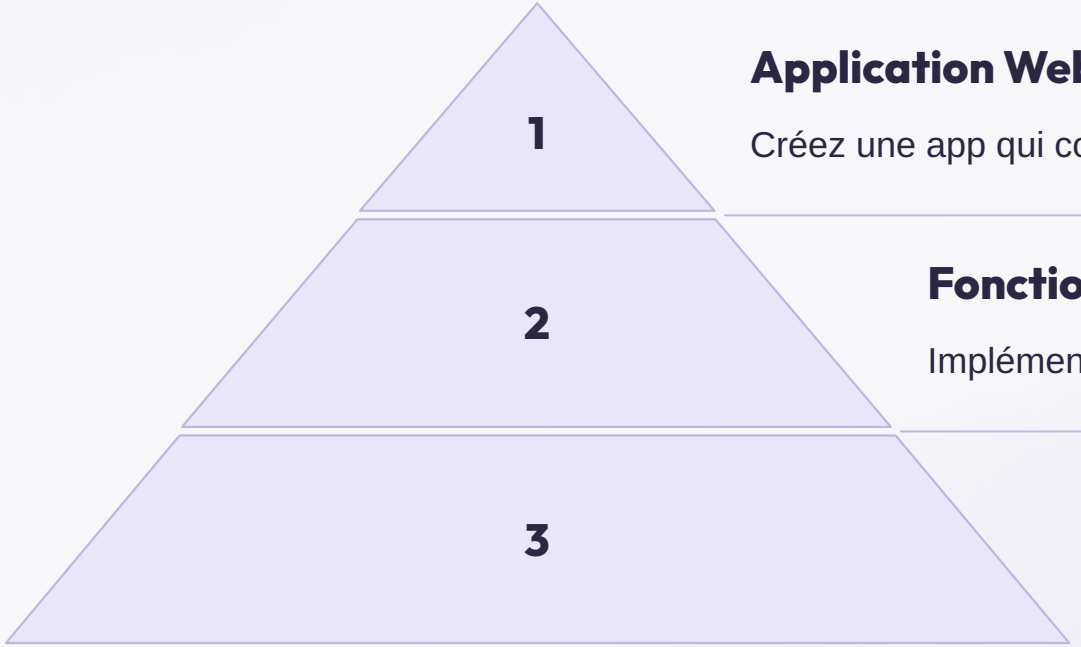
Activité

Envoyer une requête avec un en-tête `Authorization`

Utiliser un API nécessitant un jeton d'authentification



Conclusion et Projet Final



Application Web

Créez une app qui consomme une API externe.

Fonctionnalités CRUD

Implémentez l'ajout, la modification et la suppression d'articles.

Évaluation

Testez l'application avec différentes API.

Ressources Supplémentaires

Explorez la documentation officielle de fetch sur MDN, les tutoriels sur JavaScript.info, et pratiquez sur Exercism et Codewars pour approfondir vos connaissances.

Documentation officielle de `fetch`: **MDN Fetch API**

Documentation officielle `Méthodes de requête Http`: **Méthodes de requête HTTP**

Tutoriels complémentaires sur **JavaScript.info**.

Plateformes d'exercice pour pratiquer : **Exercism**, **Codewars**.