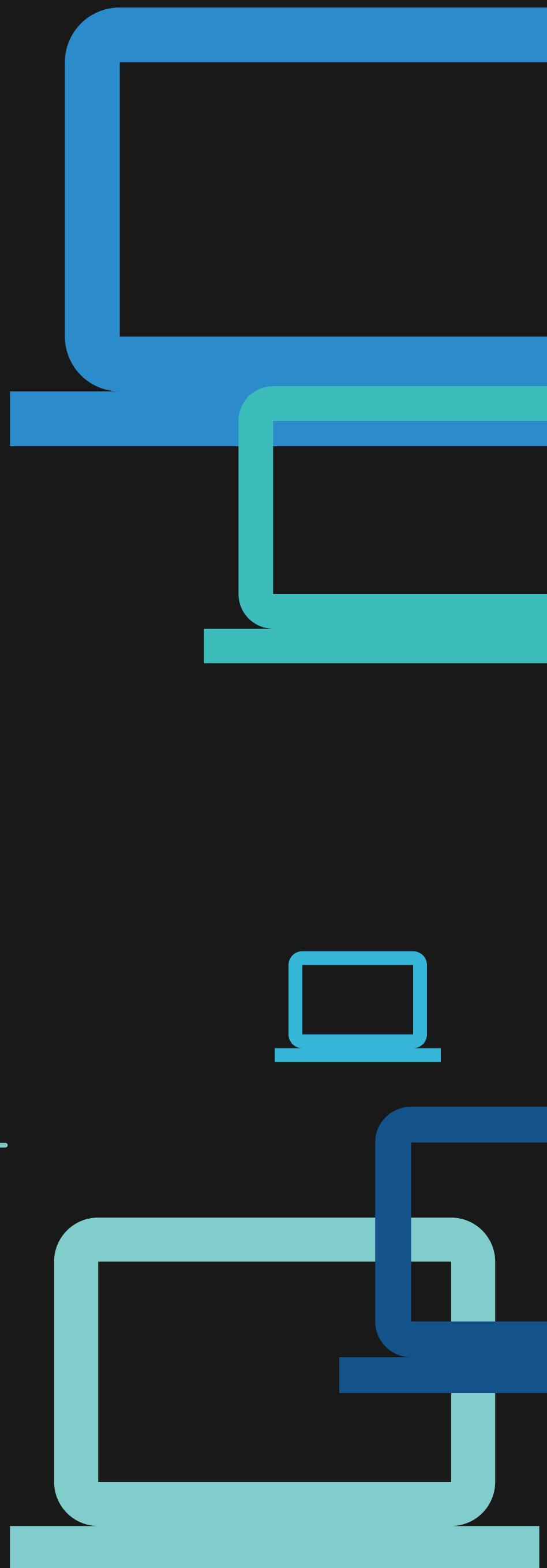


10

Fetch



Fetch

L'API Fetch est une interface JavaScript moderne pour effectuer des requêtes réseau, généralement vers des ressources sur le Web (comme des API REST). Elle remplace progressivement l'ancienne API XMLHttpRequest pour les requêtes AJAX en raison de sa simplicité et de sa facilité d'utilisation.

Vous créez une requête avec `fetch()` en spécifiant l'URL de la ressource à laquelle vous souhaitez accéder.

Fetch utilise des promesses pour gérer les réponses asynchrones. Vous utilisez `.then()` pour gérer la réponse et `.catch()` pour gérer les erreurs.

L'utilisation de Fetch permet d'aérer le code et se concentrer uniquement sur la requête et son traitement.

```
fetch('https://api.example.com/data')
  .then(response => {
    // Traitez la réponse ici
  })
  .catch(error => {
    // Gérez les erreurs ici
});
```

Envoyer une requête GET et POST

Par défaut, `fetch()` effectue une requête GET. Vous pouvez spécifier d'autres méthodes HTTP comme POST, PUT, DELETE, etc., en incluant un objet de configuration.

Options de Configuration

- `method`: La méthode HTTP de la requête (GET, POST, PUT, DELETE, etc.).
- `headers`: Un objet contentant les en-têtes HTTP de la requête, généralement utilisés pour spécifier le type de contenu ou les informations d'authentification.
- `body`: Le corps de la requête, utilisé pour envoyer des données avec les requêtes POST ou PUT, généralement au format JSON ou `FormData`.

```
const newUser = {  
  nom: nom,  
  email: email,  
  age: age  
};  
  
const config = {  
  method: 'POST',  
  headers: {  
    'Content-Type': 'application/json'  
  },  
  body: JSON.stringify(newUser)  
  
  // Autres options de configuration  
}  
  
fetch('https://api.example.com/data', config)  
  .then(response => {  
    // Traitez la réponse ici  
  })  
  .catch(error => {  
    // Gérez les erreurs ici  
});
```

Traitement de la réponse

La réponse renvoyée par `fetch()` est un objet `Response`. Vous pouvez extraire différentes informations de la réponse, telles que le statut HTTP, les en-têtes et le corps de la réponse.

Selon le type de contenu de la réponse, vous utiliserez différentes méthodes pour extraire les données. Par exemple, `.json()` pour JSON, `.text()` pour du texte brut, et ainsi de suite. Ces fonctions retournent à leur tour une promesse donc vous devez ajouter un second `.then` pour récupérer la données ainsi traitée.

Autre propriété intéressante

- `Response.ok` : Cette propriété renvoie `true` si le code de statut HTTP est compris entre 200 et 299, ce qui indique que la requête a réussi.

```
fetch('https://api.example.com/data')
  .then(response => {
    if (response.ok) {
      // Récupérer le code de réponse
      console.log(response.status)

      // Statut HTTP 200-299
      // Pour extraire le corps JSON de la réponse
      return response.json();
    } else {
      throw new Error('Réponse non valide');
    }
  })
  .then(data => {
    // Traitez les données ici
  })
  .catch(error => {
    // Gérez les erreurs ici
  });
}
```

Exemple complet

Voici un exemple concret de requête POST avec Fetch dans un contexte réaliste où vous créez un nouvel utilisateur en envoyant des données JSON à un serveur.

Supposons que vous ayez un formulaire HTML pour créer un nouvel utilisateur avec des champs tels que le nom, l'email et l'âge. Vous souhaitez utiliser Fetch pour envoyer ces données au serveur sous forme de requête POST au format JSON.

Comme les données seront envoyées au format Json, du côté du serveur, vous pourrez récupérer les données avec `file_get_contents("php://input")`; (Voir les notes du cours 9)

```
// Sélectionnez le formulaire HTML
const userForm = document.getElementById('userForm');

// Écoutez l'événement de soumission du formulaire
userForm.addEventListener('submit', (e) => {
    e.preventDefault(); // Empêche la soumission du formulaire par défaut

    // Récupérez les valeurs des champs du formulaire
    const nom = document.getElementById('nom').value;
    const email = document.getElementById('email').value;
    const age = document.getElementById('age').value;

    // Créez un objet contenant les données de l'utilisateur
    const newUser = {
        nom,
        email,
        age
    };

    // Effectuez la requête POST avec Fetch
    // Indiquez que vous envoyez des données JSON
    // Convertissez l'élément en chaîne JSON avant de le passer au body
    fetch('https://api.example.com/users', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(newUser)
    })
    .then(response => {
        if (response.ok) {
            // Traitez la réponse JSON si la requête réussit
            return response.json();
        } else {
            throw new Error('La création de l\'utilisateur a échoué.');
        }
    })
    .then(data => {
        console.log('Utilisateur créé avec succès :', data);
        // Traitez la réponse du serveur ici
        // Appeler une fonction qui met à jour les éléments HTML
    })
    .catch(error => {
        console.error('Erreur lors de la création de l\'utilisateur :', error);
        // Gérez les erreurs ici
        // par exemple, affichez un message d'erreur à l'utilisateur
    });
});
```

Références

MDN - Fetch

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

MDN - Réponse Fetch

<https://developer.mozilla.org/en-US/docs/Web/API/Response>

ColorCode - Fetch

<https://www.youtube.com/watch?v=ubw2hdQlI4E&t=617s>