4 minutes

Les objets FormData

L'objet `FormData`

`FormData` est un objet natif, il s'agit d'une `Web API` des navigateurs, qui permet d'envoyer un ensemble de paires clé / valeur et notamment des fichiers>.

Il permet essentiellement d'envoyer des données de formulaire, mais il est également possible de l'utiliser sans formulaire.

Les données sont envoyées au format `multipart/form-data` et l'entête `Content-Type` soit contenir `multipart/form-data`.

Le format `multipart/form-data`

Ce format est utilisé pour les formulaires qui envoient **des fichiers, des données non `ASCII` et des données binaires**.

`multipart` est un type `MIME` signifiant littéralement plusieur parties.

En utilisant le `media-type multipart/form-data`, chaque partie contient un `header` et un `body`.

Le `header` contient notamment `Content-Disposition: form-data;` pour indiquer au serveur le début d'un message de type `form-data`, il contient également le `name` unique du champ du formulaire.

Le `body` peut contenir tout type de média (vidéo, audio, images etc) et peut contenir du texte ou des données binaires.

`form-date` : en utilisant cet encodage, le navigateur va créer un message `multipart` pour chaque champ du formulaire.

Voici un exemple de message `multipart/form-data` transmis à un serveur :

```
…
Content-Type: multipart/form-data; boundary=---------------------------9051914041544843365972754266
Content-Length: 554

-----------------------------9051914041544843365972754266
Content-Disposition: form-data; name="text"

text default
```

```
----------------------------905191404154484336597275426 6
Content-Disposition: form-data; name="file1"; filename="a.txt"
Content-Type: text/plain

Content of a.txt.

----------------------------905191404154484336597275426 6
Content-Disposition: form-data; name="file2"; filename="a.html"
Content-Type: text/html

<!DOCTYPE html><title>Content of a.html.</title>

----------------------------905191404154484336597275426 6--
```

Vous pouvez observer qu'une `boundary` est choisie par le navigateur pour séparer chaque message.

Chaque message a bien ensuite le `header` que nous avons vu avec le nom du champ.

Ensuite le `body` contient les données qui peuvent être en binaire, en HTML, en text/plain etc.

Le navigateur va essayer de déterminer le type envoyé pour chaque champ.

Utilisation de `FormData`

Nous allons prendre un exemple pour comprendre le fonctionnement de `FormData`.
https://codesandbox.io/embed/js-c15-l7-1-9njo3

Vous pouvez remarquer que l'objet `FormData` est itérable car nous pouvons utiliser une boucle `for...of` dessus.

Vous remarquez également qu'il contient des paires de clé / valeur où la clé est l'attribut `name` du champ côté `HTML` et où la valeur est la valeur du champ.

Envoyer des données `FormData`

Pour envoyer un objet `FormData` rien de plus simple avec `fetch()` :

```
form.onsubmit = e => {
  e.preventDefault();
  const reponse = await fetch(URL, {
      method: 'POST',
      body: new FormData(form)
```

```
    });
};
```

`fetch` va automatiquement détecter qu'il s'agit de données `FormData`. Il va automatiquement encoder les données au format `multipart/form-data` (ajouter les bons entêtes etc), vous n'avez rien à faire !

Les méthodes disponibles

Il existe plusieurs méthodes permettant de manipuler des `FormData` :

`formData.append(nom, valeur)` : permet d'ajouter un champ à l'objet avec le nom et la valeur indiqués.

`formData.set(nom, valeur)` : permet d'ajouter un champ à l'objet avec le nom et la valeur indiqués et de supprimer tous les autres champs avec le même nom. Cela permet de s'assurer qu'il n'y a aucun autre champ avec le même nom sur l'objet.

`formData.delete(nom)` : permet de supprimer le champ sur l'objet ayant le nom indiqué.

`formData.get(nom)` : permet de récupérer le champ sur l'objet ayant le nom indiqué.

`formData.has)` : permet de vérifier que le champ sur l'objet ayant le nom indiqué est présent, la méthode retourne un booléen.

`formData.append(nom, blob, nomDeFichier)` : permet d'ajouter un champ fichier à l'objet avec des données passées au format `blob` (nous l'étudierons) et de spécifier le nom du fichier.

`formData.set(nom, blob, nomDeFichier)` : permet de définir la valeur du champ fichier ayant le nom indiqué sur l'objet avec des données passées au format `blob` et de spécifier le nom du fichier.

Envoyer des fichiers avec `FormData`

Pour envoyer un objet `FormData` contenant des fichiers, c'est très simple également.

Côté `HTML` :

```
<form id="monForm">
  <input type="text" name="prenom" value="Paul">
  <input type="file" name="fichier">
  <input type="submit">
</form>
```

Côté `JavaScript`, aucune différence :

```
monForm.onsubmit = e => {
  e.preventDefault();
  const reponse = await fetch(URL, {
      method: 'POST',
      body: new FormData(form)
    });
};
```

Voici un exemple :
https://codesandbox.io/embed/js-c15-l7-2-oejvk