

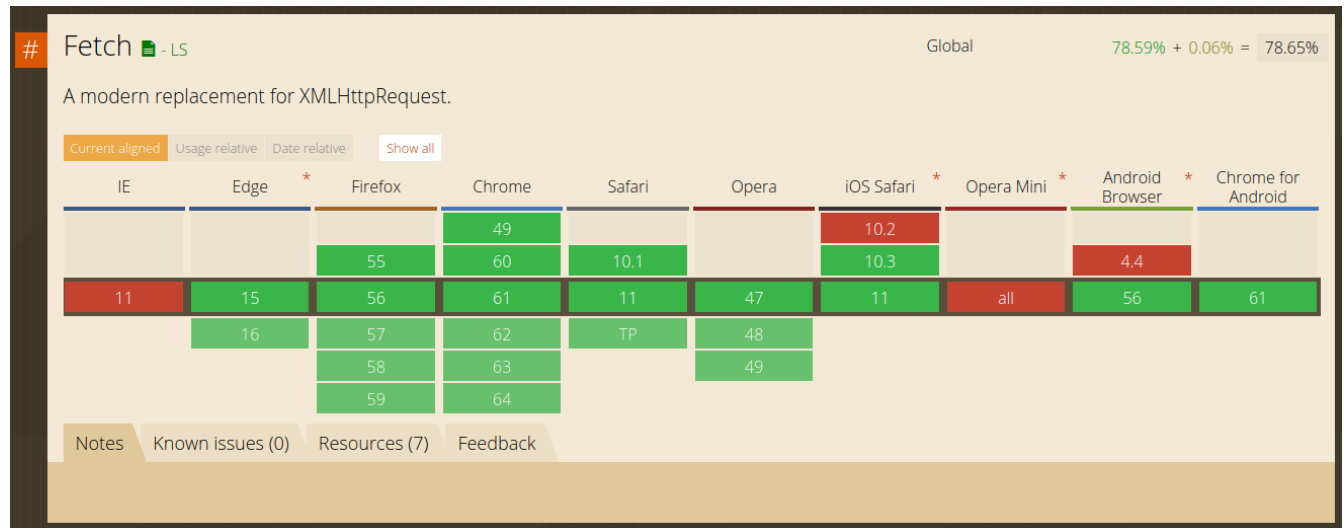
Formation Programmation Multi-Plateforme

TP - Fetch API

Présentation

Fetch API est le remplaçant moderne de *XMLHttpRequest*.

Support navigateur



```
if ('fetch' in window) {  
  // ok  
}
```

Il est possible d'utiliser un *polyfill* (<https://github.com/github/fetch>).

Exemple de requête :

```
fetch('data/data.json')  
  .then(function(response) {  
    // traiter la réponse  
  })  
  .catch(function(error) {  
    console.log('Il semble avoir un soucis...', error);  
  });
```

Objet Response

L'objet *Response* représente la réponse de la requête.

Statut de la réponse

Pour connaître le statut de la réponse :

```
response.status // code de la réponse

response.statusText // libellé du code de la réponse

response.ok // si le code de la réponse est compris entre 200 et 299.
```

Lire le contenu de la réponse

```
response.json() // récupérer un objet JS

response.blob() // récupérer une ressource binaire (images par exemple)

response.text() // récupérer du texte
```

- Exemple avec un objet JSON

```
fetch('data/superdata.json')
  .then(function(response) {
    if (!response.ok) {
      throw Error(response.statusText);
    }
    // lecture du corps de la réponse en tant que JSON.
    return response.json();
  })
  .then(function(responseAsJson) {
    // traitement de l'objet
    console.log(responseAsJson);
  })
  .catch(function(error) {
    console.log('Une erreur est survenue : ', error);
  });
```

- Exemple avec une image

```

fetch('data/superimage.png')
.then(function(response) {
  // lecture du corps de la réponse en tant que Blob.
  return response.blob();
})
.then(function(responseAsBlob) {
  // dans l'hypothèse qu'une <div> existe avec l'id = 'container'
  var container = document.getElementById('container');
  var imgElem = document.createElement('img');
  container.appendChild(imgElem);
  var imgUrl = URL.createObjectURL(responseAsBlob);
  imgElem.src = imgUrl;
})
.catch(function(error) {
  console.log('Une erreur est survenue : ', error);
});

```

Personnaliser la requête

- L'attribut *method* permet de spécifier la méthode *HTTP*.

```

fetch('data/supertexte.txt', {
  method: 'HEAD'
})

```

- L'attribut *body* spécifie le corps de la requête.

```

fetch('data/supercommentaire', {
  method: 'POST',
  body: 'title=cool&message=voyagecool'
})

```

- L'attribut *headers* permet de spécifier des entêtes.

```

// POST
fetch('data/supercommentaire', {
  method: 'POST',
  headers: {
    "Content-Type": "application/json"
  },
  body: '{"data": "content"}'
})

```

Pour aller plus loin

- <https://fetch.spec.whatwg.org/>
- https://developer.mozilla.org/fr/docs/Web/API/Fetch_API/Using_Fetch

Travaux Pratiques

Installation

- Créer un répertoire *01-fetch-api* :

```
mkdir 01-fetch-api
```

- Initialiser un projet *npm* :

```
npm init -y
```

- Installer un serveur :

```
npm i http-server -D
```

- Compléter le fichier *package.json* pour ajouter la tâche *start* :

package.json

```
{  
  ...  
  "scripts": {  
    "start" : "http-server -a localhost -c 0",  
    ...  
  },  
  ...  
}
```

- Démarrer le serveur :

```
npm start
```

Liste des sessions

- Ajouter une page Web *sessions.html* :

```

<!doctype html>
<html>
  <head>

  </head>
  <body>
    <h1>Liste de sessions</h1>

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

```

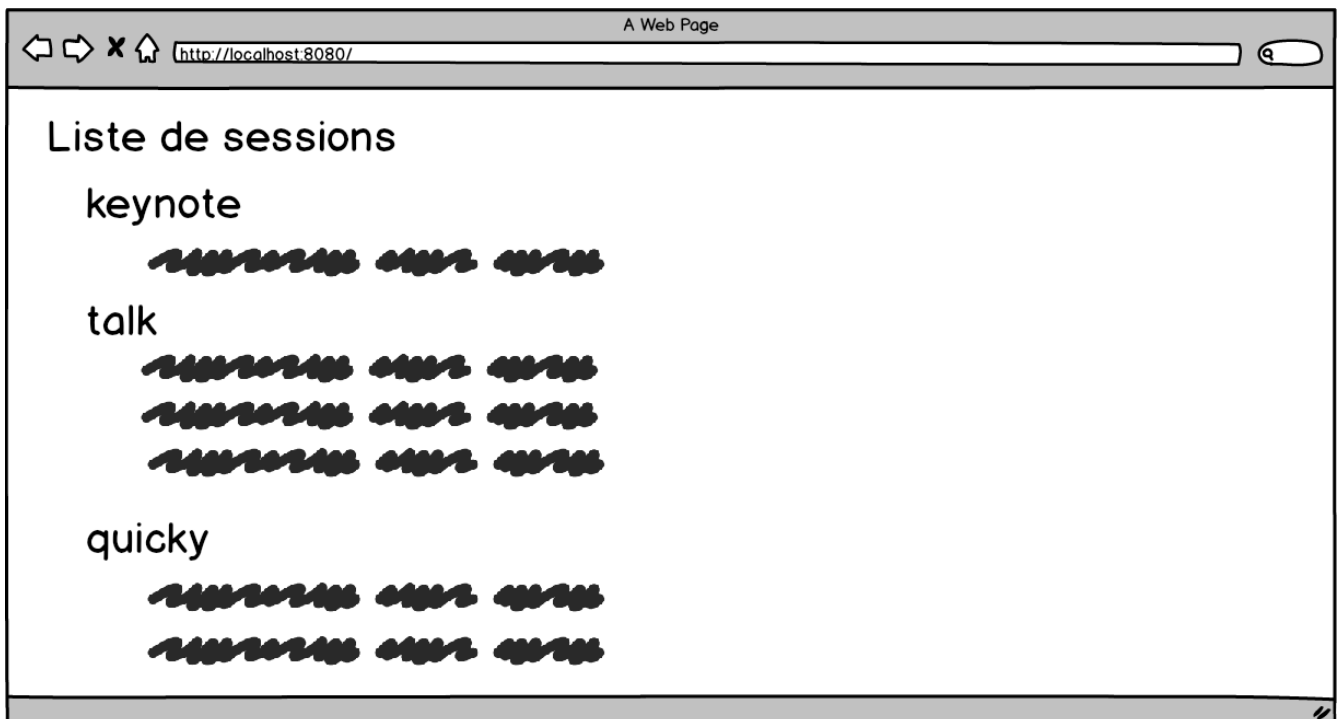
- Ajouter un fichier *sessions.js* vide. A ce stade, l'arborescence est la suivante :

```

/01-fetch-api
  sessions.html
  sessions.js
  package.json
  /node_modules/

```

- Implémenter la page de liste de sessions pour qu'elle affiche :
 - toutes les sessions classées par catégorie (propriété *type*)
 - pour chaque session, afficher le titre de la session suivi des noms et prénoms des présentateurs.



Gestion des notes

- Ajouter une page Web *notes.html* :

sessions.html

```
<!doctype html>
<html>
  <head>

  </head>
  <body>
    <h1>Gestion des notes</h1>

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

- Ajouter un fichier *notes.js* vide. A ce stade, l'arborescence est la suivante :

```
/01-fetch-api
  sessions.html
  sessions.js
  notes.html
  notes.js
  package.json
  /node_modules/
```

- Installer *json-server* :

```
npm i -D json-server
```

- Créer un fichier *data.json* :

data.json

```
{
  "notes": []
}
```

- Compléter le fichier *package.json* :

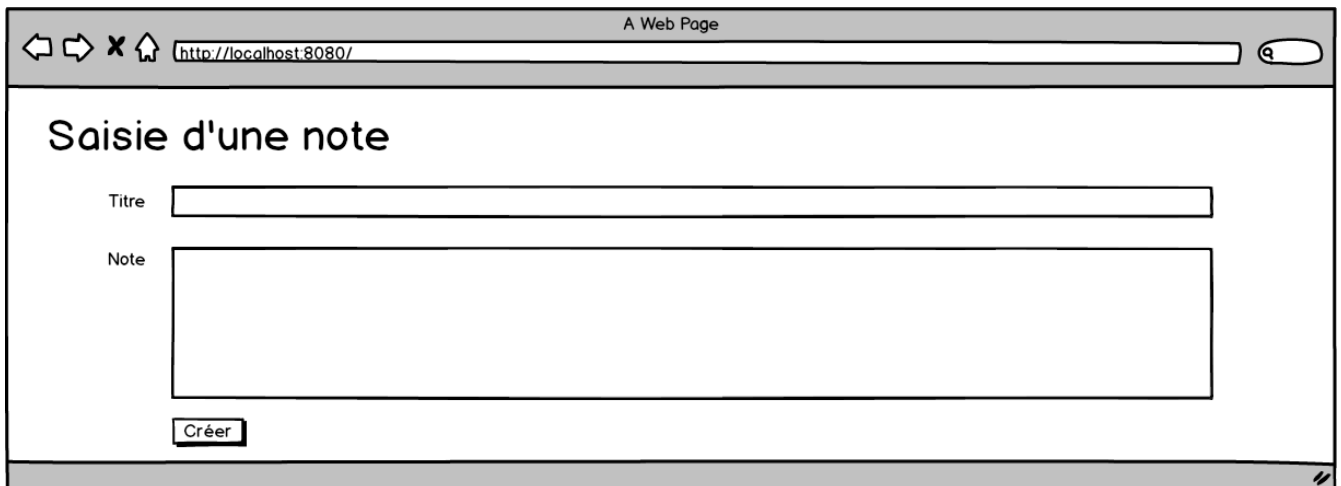
package.json

```
{  
  ...  
  "scripts": {  
    ...  
    "fake:api" : "json-server --watch data.json",  
    ...  
  }  
  ...  
}
```

- Démarrer le serveur :

```
npm run fake:api
```

- Implémenter un formulaire de création d'une note :



The screenshot shows a web browser window with the title 'A Web Page'. The address bar displays 'http://localhost:8080/'. The main content area has the heading 'Saisie d'une note'. Below the heading, there are two input fields: 'Titre' (a single-line text box) and 'Note' (a multi-line text area). At the bottom left of the form, there is a button labeled 'Créer'.

Un clic sur le bouton *Créer* sauvegarde une note dans *json-server*.