DEV

#CestFacile : Fetch ou Axios ?

Publié par DAIBAI *le* 2 MARS 2018

(oui, c'est vrai, la blague est éculée, mais j'aime bien)

Vous connaissez sûrement le bon vieux $.ajax issu de jQuery ? Hey ! Y a pas de mal hein, ce n'est pas sale. Je l'ai utilisé des milliers de fois à l'époque (et encore de nos jours, parfois), et c'était bien pratique.

Dans une application React, on va plutôt utiliser Fetch, Axios, ou SuperAgent, enfin… on va surtout éviter d'utiliser jQuery.

Allez, on se fait une (toute) petite comparaison entre Fetch et Axios.

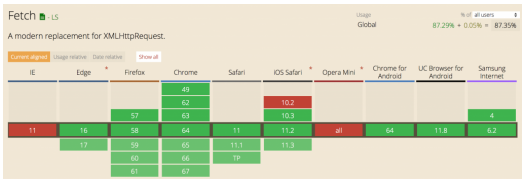
Fetch

DEV

#CestFacile : Fetch ou Axios ?

Publié par DAIBAI *le* 2 MARS 2018

Avant tout, l'API Fetch a le mérite d'exister nativement dans la plupart des navigateurs :



C'est le « *dans la plupart des navigateurs* » le soucis. Par exemple Safari 10.2 et IE 11 vous enverront bouler, ce qui signifie que vous devrez utiliser un polyfill pour pouvoir utiliser Fetch sereinement.

L'idée autour d'un polyfill est de rendre une API compatible avec la majorité des navigateurs. C'est une bonne démarche car en admettant que cette API soit un jour compatible avec tous les navigateurs, ça ne coute pas grand chose de virer le polyfill qui ne servira plus à rien.

Une requête GET sur une API avec Fetch, ça ressemble à ça :

```
fetch('https://api.chuc
  .then((response) => {
    return response.jso
  })
  .then( (data) => {
```

DEV

#CestFacile : Fetch ou Axios ?

Publié par DAIBAI *le* 2 MARS 2018

```
    return data;
  })
  .catch((err) =>{
    console.log(err)
  })
```

On doit transformer notre response en json (response.json()) pour pouvoir exploiter les données issues des webservices, basé sur les promises.

On peut bien sûr ajouter des options, configurer les headers (pour y passer un token JWT par exemple), même si ça peut devenir à l'occasion un peu chiant et un poil verbeux.

```
fetch('https://ce_que_v
  method: 'POST',
  headers: {
    'Authorization': 'B
    'Content-Type': 'ap
  },
  body: 'firstName=Bobb
})
  .then((response) => {
  .then( (data) => { re
  .catch((err) =>{ cons
```

Dernier point, la gestion des erreurs peut devenir un poil casse bonbon dans certains cas.

DEV

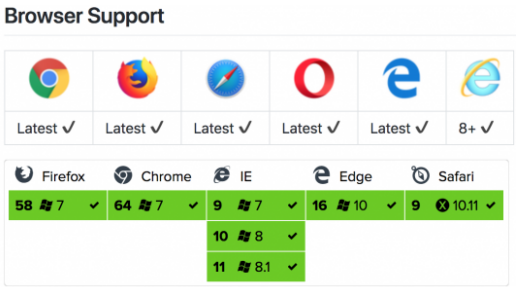
#CestFacile : Fetch ou Axios ?

Publié par DAIBAI *le* 2 MARS 2018

Quoiqu'il en soit Fetch est un excellent choix, et c'est une solution pérenne lorsque l'on utilise un polyfill.

Axios

Axios reprend les mêmes concepts mais avec un support étendu à toutes les versions de navigateurs (y compris IE 8)



Entre autres avantages :

- Axios s'occupe automatiquement de transformer les réponses en JSON (exit le response.json())
- Il est possible d'annuler une requête
- le catch() semble poser nettement moins de soucis qu'avec Fetch

Une requête GET avec Axios, ça ressemble à ça :

DEV

#CestFacile : Fetch ou Axios ?

Publié par DAIBAI *le* 2 MARS 2018

```
axios.get('https://api.
  .then(function (respo
    return response;
  })
  .catch(function (erro
    console.log(error);
  });
```

Pas un changement dramatique par rapport à Fetch.

Pareil pour un POST :

```
axios.post('https://ce_
    firstName: 'Bobbix'
    lastName: 'Jean'
  }, {
    headers:{
      'Authorization':
      'Content-Type': '
    }
  })
  .then(function (respo
    console.log(respons
  })
  .catch(function (erro
    console.log(error);
  });
```

Truc intéressant, vous pouvez passer plusieurs requêtes à Axios comme vous le feriez avec un promise.all(), c'est

DEV

#CestFacile : Fetch ou Axios ?

Publié par DAIBAI *le* 2 MARS 2018

exactement le même principe (issu de la doc d'Axios) :

```
function getUserAccount
  return axios.get('/us
}

function getUserPermiss
  return axios.get('/us
}

axios.all([getUserAccou
  .then(axios.spread(fu
    // Both requests ar
  }));
```

Dans nos prochaines aventures, on partira sur Axios, mais encore une fois vous pouvez parfaitement vous en passer et préférer Fetch.

ARTICLE PRÉCÉDENT
#CestFacile :Vérification de types (feat PropTypes)

ARTICLE SUIVANT
#CestFacile : React Error Boundaries