openmrs-esm-api

What is this?

openmrs-esm-api is an in-browser javascript module that exports functions that interact with the OpenMRS API.

How do I use it?

```
import { openmrsFetch, openmrsObservableFetch, getCurrentUser, fhir } from '@openmrs/esm-api';

openmrsFetch('/ws/rest/v1/session').then(response => {
  console.log(response.data.authenticated)
}
```

Contributing / Development

Instructions for local development

API

The following functions are exported from the @openmrs/esm-api module:

openmrsFetch(url, init): Promise

The openmrsFetch function is a wrapper around the browser's built-in fetch function, with extra handling for OpenMRS-specific API behaviors, such as request headers, authentication, authorization, and the API urls.

Arguments

1. url (required): A string url to make the request to. Note that the openmrs base url (by default "/openmrs") will be automatically prepended to the URL, so there is no need to include it.
2. init (optional): A fetch init object. Note that the `body` property does not need to be JSON.stringify()'ed because openmrsFetch will do that for you.

Return value

A Promise that resolves with a Response object. Note that the openmrs version of the Response object has already downloaded the HTTP response body as json, and has an additional `data` property with the HTTP response json as a javascript object.

Example

```
import { openmrsFetch } from '@openmrs/esm-api'

const abortController = new AbortController();

openmrsFetch('/ws/rest/v1/session', {signal: abortController.signal})
  .then(response => {
    console.log(response.data.authenticated)
  })
  .catch(err => {
    console.error(err.status);
  })

abortController.abort();

openmrsFetch('/ws/rest/v1/session', {
  method: 'POST',
  body: {
    username: 'hi',
    password: 'there',
  }
})
```

Cancellation

To cancel a network request, use an AbortController. It is best practice to cancel your network requests when the user navigates away from a page while the request is pending request, to free up memory and network resources and to prevent race conditions.

openmrsObservableFetch(url, init): Observable

The openmrsObservableFetch function is a wrapper around openmrsFetch that returns an Observable instead of a promise. It exists in case using an Observable is preferred or more convenient than a promise.

Arguments

The arguments to openmrsObservableFetch are exactly the same as the arguments to openmrsFetch.

Return value

An Observable that produces exactly one Response object. The response object is exactly the same as for openmrsFetch.

Example

```
import { openmrsObservableFetch } from '@openmrs/esm-api'

const subscription = openmrsObservableFetch('/ws/rest/v1/session').subscribe(
  response => console.log(response.data),
  err => {throw err},
  () => console.log('finished')
)
subscription.unsubscribe()
```

Cancellation

To cancel the network request, simply call subscription.unsubscribe();

fhir

The `fhir` object is an instance of fhir.js that can be used to call FHIR-compliant OpenMRS APIs. See the docs for fhir.js for more info and example usage.

getCurrentUser(): Observable

The getCurrentUser function returns an observable that produces ***zero or more values, over time.*** It will produce zero values by default if the user is not logged in. And it will provide a first value when the logged in user is fetched from the server. Subsequent values will be produced whenever the user object is updated.

Arguments

1. options (optional): An object with includeAuthStatus boolean property that defaults to false. When includeAuthStatus is set to true, the entire response object from the API will be provided. When includeAuthStatus is set to false, only the user property of the response object will be provided.

Return value

An Observable that produces zero or more values (as described above). The values produced will be a user object (if includeAuthStatus is set to false) or an object with a session and authenticated property (if includeAuthStatus is set to true).

Example

```
import { getCurrentUser } from '@openmrs/esm-api'

const subscription = getCurrentUser().subscribe(
  user => console.log(user)
)
subscription.unsubscribe()

getCurrentUser({includeAuthStatus: true}).subscribe(
  data => console.log(data.authenticated)
)
```

Be sure to unsubscribe when your component unmounts

Otherwise your code will continue getting updates to the user object even after the UI component is gone from the screen. This is a memory leak and source of bugs.

refetchCurrentUser(): Observable

The refetchCurrentUser function causes a network request to redownload the user. All subscribers to the current user will be notified of the new users once the new version of the user object is downloaded.

Arguments

None

Return value

An observable exactly the same as if you had called getCurrentUser().

Example

```
import { refetchCurrentUser } from '@openmrs/esm-api'

refetchCurrentUser()
```