
Take control of your mock !
Don’t worry, I have a backup plan !

Johnathan MEUNIER

Lead front @AXA France

@MonsieurNohj

John.meunier

Table of contents

01
Introduction

My demo is broken !


gif-finder.com

My demo seems broken

How can I prevent that ?

Internet issue

Work without Internet, in a train, when my company proxy is not available

API issue

My API is down or has changed

Prepare scenarios

I don’t want to fill all my forms everytime or I want to improve the resilience of my app

02

Mocks

What is it ? What’s now ? Demo

What is a mock ?

- Simulation of what would happen in a normal state
- Not request directly the API
- Simplify response
- Have to be static (most of the time)

Pros & Cons

Pros

- Work offline
- Simulate your local dev with different scenarios
- Stable environment, whatever the state of your service
- A mock is almost instantaneous …

Cons

- … but the real behavior of your application can be distorded
- Your local data have to be up to date permanently with the real API (path, values, schema, etc)

Current solution

- Package like json-server can create a simple rest API which will serve static json file (or complex in some cases)
- Powerful, this solution will redirect all the calls to a local webserver …

- … but you need to launch a specific webserver in addition to your application
- in your application, you will have specific code to manage the switch between this local server and the real API

Current solution

- Simplier, you can create static json file with your component and use this file.
- Useful to initiate a development but impossible to use this code in production
- Really far from the real behavior of your application

Demo of my working application

03

Mock Service Worker

Setup, usage, tests, msw-ui,
scenarios, advanced usage

MSW is a service worker

- A proxy between a web app and your browser
- Sandboxed
- Lower level API

Why MSW is the best solution ?

- MSW will intercept some requests of the network layer of your browser
- Totally transparent during the development phase
- Nothing to do in the functionnal code, in your compoments !
- Every is centralized in a specific part of your codebase

Front setup

Request handler

```
import { rest } from 'msw';

http.[get|post|put|•••](url,callback)
```

- get
- post
- put
- patch
- delete
- options

- **Exact** : https://api.backend.dev/users
- **Wildcard** : users/*
- **Parameters** : `https://api.backend.dev/users/:userId`
- **RegExp** : /\/users\//

Request resolver

When the request is caught, you have to return a response. The response have to be a valid **Response** from the fetch API. So you can retrieve the base Response from the fetch API or the HttpReponse provided by MSW. **HttpReponse** make it easier some features :

- return json, xml, formData, etc
- set status
- provide headers
- mocking errors
- ...

Request resolver

A complete but simple mock looks like this :

```
1  http.get("https://the-one-api.dev/v2/character", async () ⇒
2    HttpResponse.json(getCharacter))
```

Request resolver

Add delay (always add delay !)

```
1  await delay()
```

Change status

```
1  HttpResponse({ error: "an error message" }, { status: 500 })
```

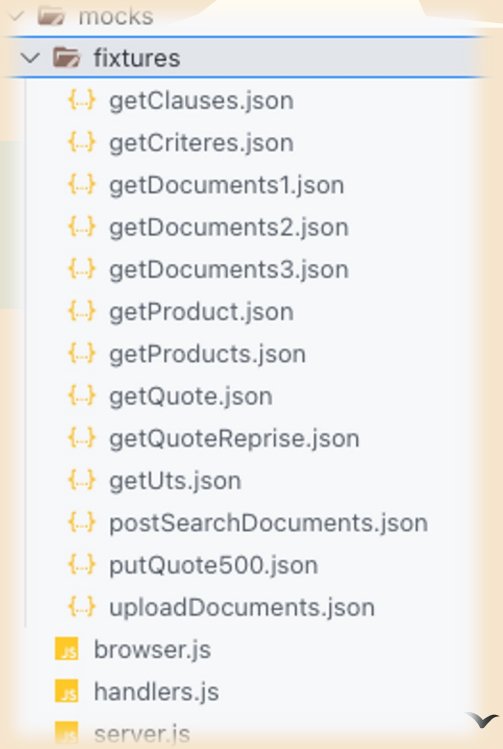
Provide headers

```
1  return HttpResponse(null,
2    { headers: { 'Set-Cookie': 'mySecret=abc-123'},
3  })
```

Request resolver

Organization

```
src/mocks/fixtures
[verbHTTP][ressource][metaInfo].json
```



Request resolver

Complex response

```
1  http.post("/documents/:documentId", async ({ request }) => {
2    const document = documents[request.params.document];
3    document.list.sort(() => Math.random() - 0.5);
4    return HttpResponse.json(document);
5  })
```

Demo : fetch, simple mock, delay, status, headers, complex response

Test

Avoid mocking directly fetch, axios or your api consumption hook because you will create a dependance.

Mock your call throught MSW, regardless of how you interrogate it.

Specific mock in a test

Simple solution

```
1  it("Should render the page correctly", async () ⇒ {
2      server.use(
3        http.get("https://the-one-api.dev/v2/character", async () ⇒ {
4          await delay(1000);
5          return HttpResponse.json(getCharacter, { status: 200 });
6        })
7      );
8      render(<LOTR />);
9      expect(screen.getByText("Loading")).toBeInTheDocument();
10     expect(await screen.findByText("Amrod")).toBeInTheDocument();
11   });
```

Specific mock in a test

With a gherkin feature file

```
given("Je cherche une liste de personnages", () ⇒ {
  server.use(
    http.get("https://the-one-api.dev/v2/character", async () ⇒ {
      await delay(1000);
      return HttpResponse.json(getCharacter, { status: 200 });
    })
  );
});
```

Specific mock in a test

Variable throught .feature

```
Feature: Limite inférieure à la franchise

    Scenario Outline: Gestion de la validation du champ franchise
    Given Un devis rempli et une offre commerciale complète sont chargés avec une franchise limitée à "<minValue>"
    And J'accède à la page Tarif en tant que Siège
    When Je renseigne pour le champ "Franchise" la valeur "<franchiseValue>"
    Then Le message d'erreur "<errorMessage>" lié à la valeur de la franchise "<status>" affichée
    Examples:
      | minValue | franchiseValue | status | errorMessage |
      | 500 | 400 | est | Montant mini: 500€ |
      | 600 | 700 | n'est pas | Montant mini: 600€ |
      | 500 | 1000.6 | est | La valeur doit être un nombre entier |
      | 600 | 700 | n'est pas | La valeur doit être un nombre entier |
      | 600 | 800.6 | est | La valeur doit être un nombre entier |

```

Specific mock in a test

```
1 const givenDevisWithFranchiseLimit = (given) ⇒ {
2   given(
3     /^Un devis rempli et une offre commerciale complète sont chargés avec une franchise limitée à
  "(.*)"$/,
4     (deductibleMinValue) ⇒ {
5       const offersCopy = cloneDeep(getOffers)
6       set(offersCopy, deductibleMinValuePath, deductibleMinValue)
7       server.use(
8         rest.post('/insurancequotes/:quoteId', (req, res, ctx) ⇒
9           res(ctx.status(200), ctx.json(getOffers))
10        ),
11        rest.get(
12          `${conf.apiEbusinessUrl}/insurance-quotes/:quoteId`,
13          (req, res, ctx) ⇒ res(ctx.status(200), ctx.json(getQuote))
14        )})}
15
```

Demo MSW & Gherkin

Name your scenarios !

```
export const handlers = {
  getCharacter200: http.get("https://the-one-api.dev/v2/character", async () ⇒ {
    await delay(1000);
    return HttpResponse.json(getCharacter, { status: 200 });
  }),
  getCharacter500: http.get("https://the-one-api.dev/v2/character", async () ⇒ {
    await delay();
    return HttpResponse(null, { status: 500 });
  }),
  getPokemon200Hateoas: http.get(`${apiBaseUrlPokemon}pokemon`, async ({ request }) ⇒ {
    /*  ...  */
    return HttpResponse.json(modifyGetPokemonAll, { status: 200 });
  }),
};
```

Name your scenarios

In a test

```
1  it("Should render the page correctly", async () ⇒ {
2    server.use(handlers.getCharacter200);
3    render(<LOTR />);
4    expect(screen.getByText("Loading")).toBeInTheDocument();
5    await waitForElementToBeRemoved(() ⇒ screen.getByText("Loading"));
6    expect(screen.getByText("Amrod")).toBeInTheDocument();
7  });
```

Name your scenarios !

```
1  export const worker =
2    setupWorker(handlers.getCharacter200, handlers.getPokemon200Hateoas);
```

How far should we go in the mocks ?

- In most of cases, serving static files is enough
- But sometimes, the service enriches the data sent
- Don’t redevelop the service in your handler, keep it as simple as possible. *If you have this data in your request, so add this data in your answer.*

Typescript

- Typescript can help you.
- Your mocked data have the same model as your web service

Develop the consumer before the service

- In some cases, in can be very useful to develop the service consumer before the service :
  - You have a back for front that transform your data
  - The back-end team is late on the project
  - …
- Your contracts between your services and your consumers can’t change

Cohabitation with authentication

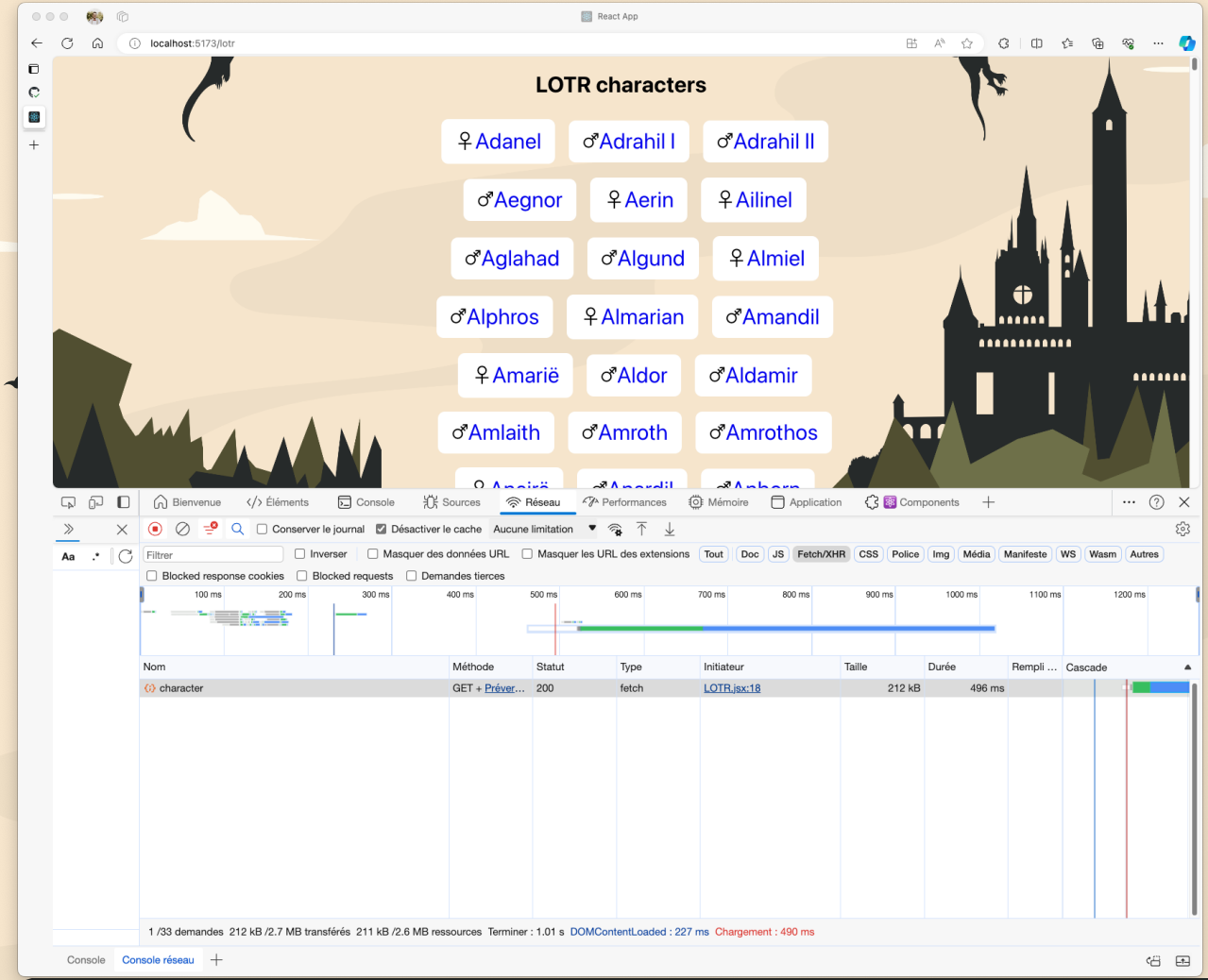
- Only one service worker per scope is allowed
- Sometimes, your authentification will work with a service worker (hello auth-worker and the latest version of react-oidc)
- During the local dev, use the MSW and mock the auth
- When you deploy, use the auth and disable mock

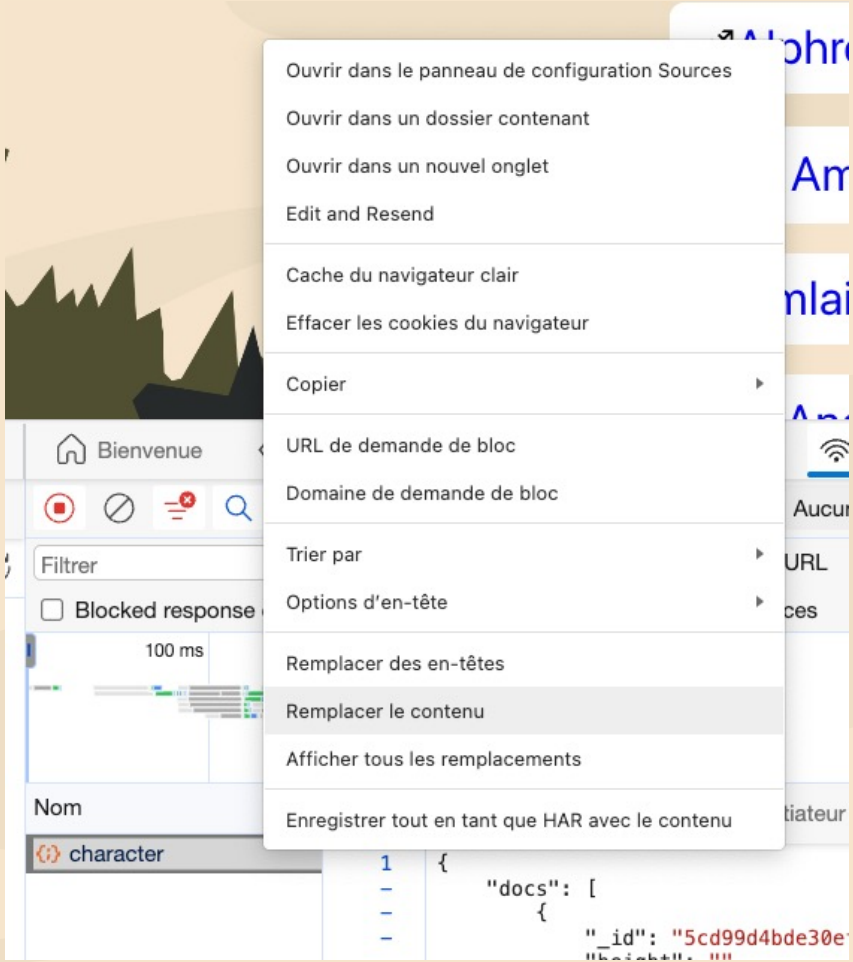
Bonus

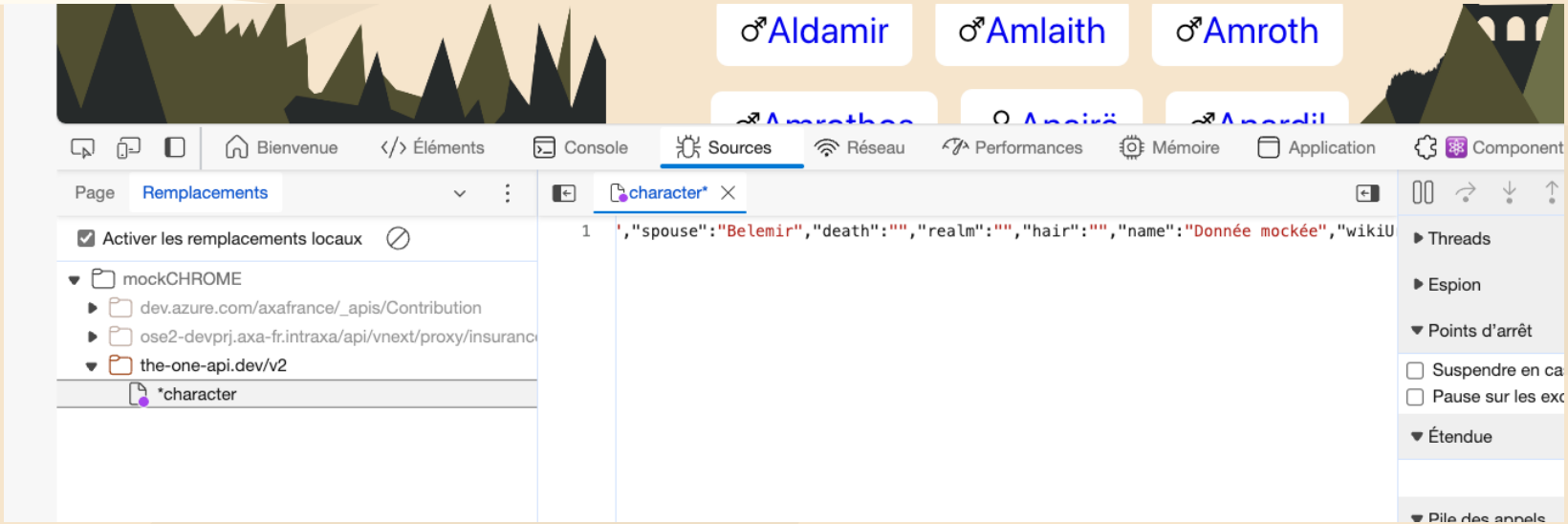
A more simple and exportable solution ?

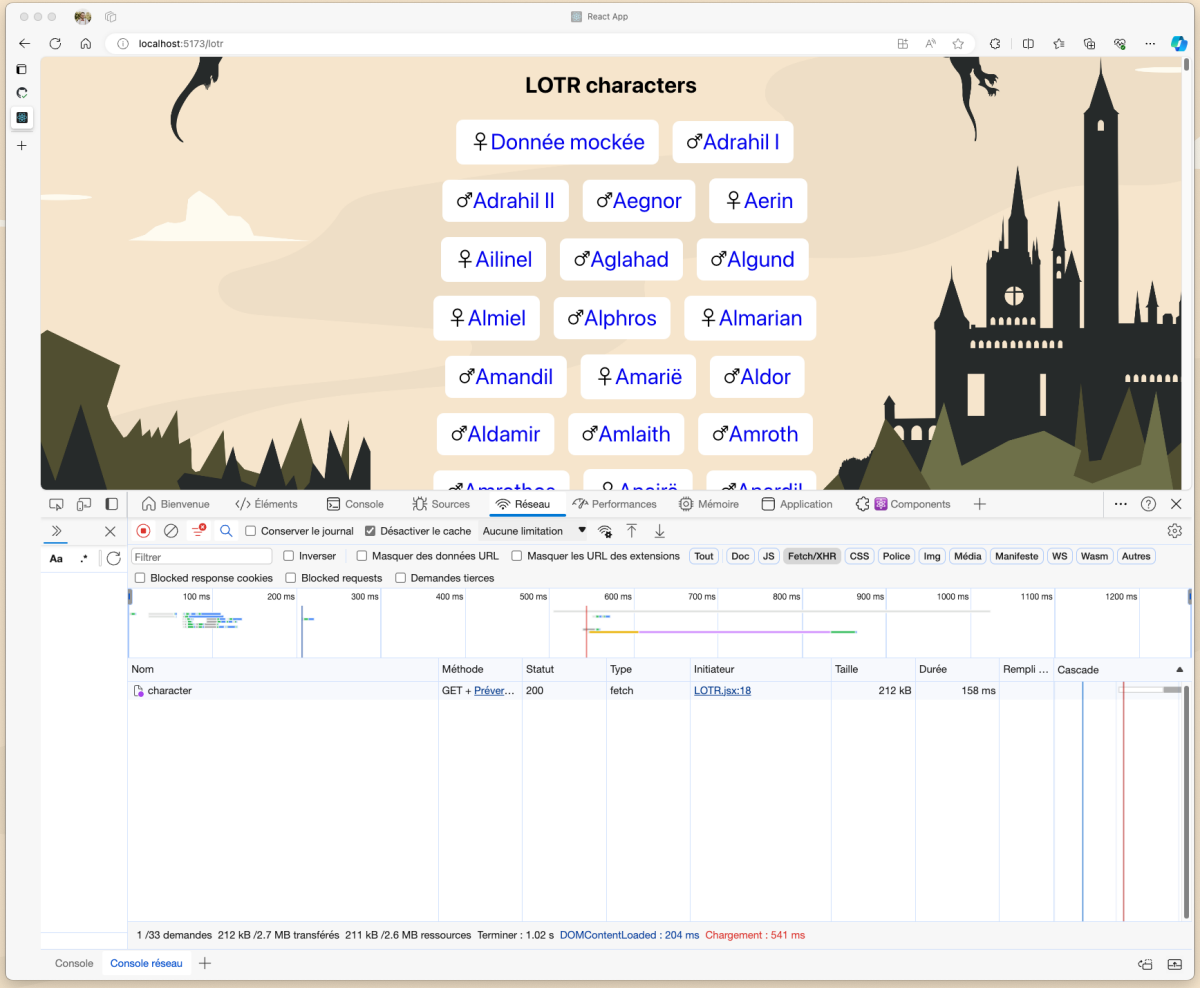
Directly from your dev tools !

- You can mock your webservices directly from your dev tools
- No setup
- Everywhere :
  - In your local env
  - In your production env
  - In any website !


React App
localhost:5173/lotr
LOTR characters
♀Adanel
♂Adrahil I
♂Adrahil II
♂Aegnor
♀Aerin
♀Ailinel
♂Aglahad
♂Algund
♀Almiel
♂Alphros
♀Almarian
♂Amandil
♀Amarië
♂Aldor
♂Aldamir
♂Amlaith
♂Amroth
♂Amrothos
Bienvenue
Éléments
Console
Sources
Réseau
Performances
Mémoire
Application
Components
Conserver le journal
Désactiver le cache
Aucune limitation
Filtrer
Inverser
Masquer des données URL
Masquer les URL des extensions
Tout
Doc
JS
Fetch/XHR
CSS
Police
Img
Média
Manifeste
WS
Wasm
Autres
Blocked response cookies
Blocked requests
Demandes tierces
100 ms
200 ms
300 ms
400 ms
500 ms
600 ms
700 ms
800 ms
900 ms
1000 ms
1100 ms
1200 ms
Nom
Méthode
Statut
Type
Initiateur
Taille
Durée
Rempli ...
Cascade
character
GET + Préver...
200
fetch
LOTR.jsx:18
212 kB
496 ms
1 /33 demandes 212 kB /2.7 MB transférés 211 kB /2.6 MB ressources Terminer : 1.01 s DOMContentLoaded : 227 ms Chargement : 490 ms
Console
Console réseau


Ouvrir dans le panneau de configuration Sources
Ouvrir dans un dossier contenant
Ouvrir dans un nouvel onglet
Edit and Resend
Cache du navigateur clair
Effacer les cookies du navigateur
Copier
URL de demande de bloc
Domaine de demande de bloc
Trier par
Options d'en-tête
Remplacer des en-têtes
Remplacer le contenu
Afficher tous les remplacements
Enregistrer tout en tant que HAR avec le contenu
Bienvenue
Filtrer
Blocked response
100 ms
Nom
character
{
"docs": [
{
"_id": "5cd99d4bde30e


♂Aldamir
♂Amlaith
♂Amroth
Bienvenue
Éléments
Console
Sources
Réseau
Performances
Mémoire
Application
Component
Page
Remplacements
Activer les remplacements locaux
mockCHROME
dev.azure.com/axafrance/_apis/Contribution
ose2-devprj.axa-fr.intraxa/api/vnext/proxy/insuranc
the-one-api.dev/v2
*character
character*
1
","spouse":"Belemir","death":"","realm":"","hair":"","name":"Donnée mockée","wikiU
Threads
Espion
Points d'arrêt
Suspendre en ca
Pause sur les exc
Étendue
Pile des appels


React App
localhost:5173/lotr
LOTR characters
♀Donnée mockée
♂Adrahil I
♂Adrahil II
♂Aegnor
♀Aerin
♀Ailinel
♂Aglahad
♂Algund
♀Almiel
♂Alphros
♀Almarian
♂Amandil
♀Amarië
♂Aldor
♂Aldamir
♂Amlaith
♂Amroth
Bienvenue
Éléments
Console
Sources
Réseau
Performances
Mémoire
Application
Components
Conserver le journal
Désactiver le cache
Aucune limitation
Filtrer
Inverser
Masquer des données URL
Masquer les URL des extensions
Tout
Doc
JS
Fetch/XHR
CSS
Police
Img
Média
Manifeste
WS
Wasm
Autres
Blocked response cookies
Blocked requests
Demandes tierces
100 ms
200 ms
300 ms
400 ms
500 ms
600 ms
700 ms
800 ms
900 ms
1000 ms
1100 ms
1200 ms
Nom
Méthode
Statut
Type
Initiateur
Taille
Durée
Rempli ...
Cascade
character
GET + Préver...
200
fetch
LOTR.jsx:18
212 kB
158 ms
1 /33 demandes 212 kB /2.7 MB transférés 211 kB /2.6 MB ressources Terminer : 1.02 s DOMContentLoaded : 204 ms Chargement : 541 ms
Console
Console réseau

04

Conclusion

Ressources, annexes, thanks


Typescript
MSW-UI
Delay
Request resolver
Service worker
Request handler
MSW
Enrichments
Scenarios
In advance
Test
Fixtures
Resilience

Thanks

Do you have any questions?

https://bit.ly/mock-msw

Johnathan MEUNIER
@MonsieurNohj

Resources

- Slide template from Slidesgo
- Repo github : johnmeunier/dojo-msw (github.com)
- MSW : MSW – Seamless API mocking library for browser and Node | Mock Service Worker (mswjs.io)
- MSW-UI : fvanwijk/msw-ui: UI for Mock Service Worker to set mock scenarios on runtime (github.com)
- AXA France GuilDev on Github (auth-worker, react-oidc, etc)