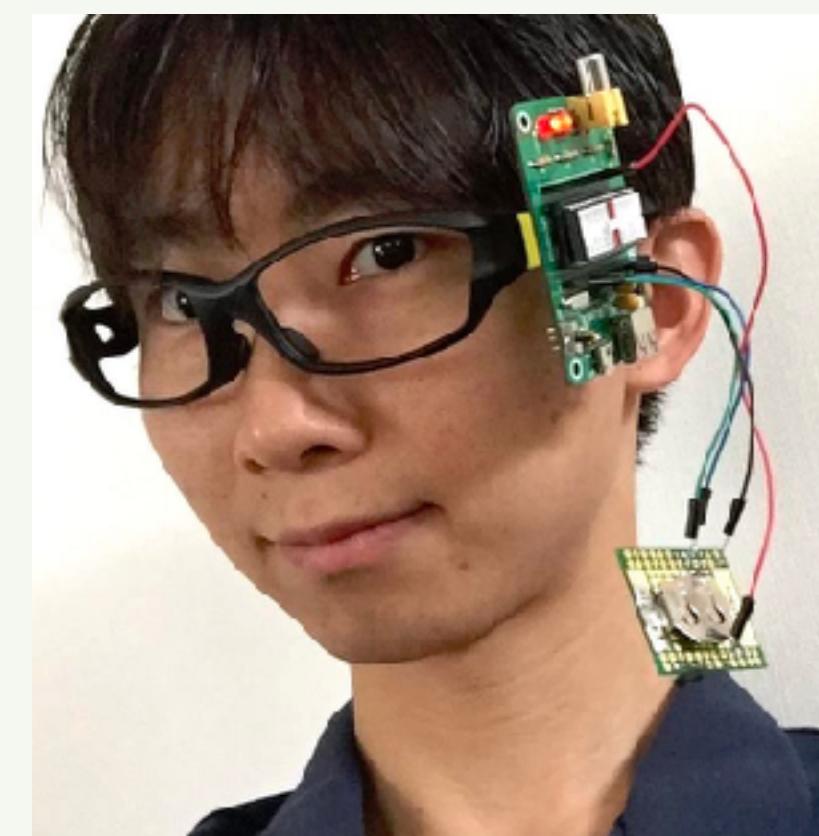


Webアプリ勉強会

超かんたんフロント × サーバー編

#jigintern 2021.2.25



ふくっち
@taisukef

劍 柄

1/30. 鮎江トイレマップ (map, data, sabae, local)	1/31. 鮎江トイレ徒步ナビ (map, sabae, local, tool)	2/3. Dataシティさばえ観光ナビ (sabae, local, data, map, tour)	2/5. さばえランチスロット (game, local, data)	2/7. 釜めしスロット (tool, local, sabae)	2/8. さばえ迷路ナビ (sabae, local, data, map, emergency)	2/14. さばえAED検索 (sabae, local, data, map, emergency)	2/22. 2.22記念さばにゃんスロット (game, canvas, sabae)
2/23. さばえ無線LANマップ (sabae, local, map)	3/3. 鮎江百景をコンテンツ化 (sabae, local, tour)	3/7. さばえ検定2012アプリ (sabae, local, game)	3/28. さばえトイレ情報のRDF化 (data, sabae, local)	4/2. さばえの人口推移をグラフ化 (data, sabae, local, opengovjp)	4/4. さばえトイレ情報マップの OSM版 (map, sabae, local, dev, ydn)	4/10. 鮎江トイレマップBing版 (sabae, local, map, dev)	4/28. 地球と鮎江の温暖化調べ (sabae, opengovjp)
5/7. SABAEグルめぐり (sabae, content)	5/11. 眼鏡スロット (game, megane, content, sabae)	5/22. 鮎江パリアフリートイレ検索 (sabae, opengovjp, map)	5/28. 精米所ナビ(福井版) (map, fukui, sabae)	6/11. 鮎江カード (content, sabae)	6/13. 鮎江市議会USTハイライト (opengovjp, content, sabae)	6/14. 鮎江グルメマップ (content, sabae, map)	6/16. さばめぐり (content, sabae, game)
6/17. 距離ビジュアライズ (content, sabae, visualize)	6/21. 台風情報巡回ツール(福井版) (content, tool, sabae, fukui)	6/22. たんなんニュース (content, sabae, echizen)	6/30. 超能力ゲーム (game, sabae)	7/16. 鮎江市廢棄量統計 (sabae, local, opengovjp)	7/23. フォーカスクイズ (sabae, local, opengovjp)	8/8. 鮎江15パズル (game, sabae)	8/16. 鮎江イベント (content, sabae)
8/17. 中国語スイーツクイズ鮎江編 (game, chinago, sabae)	8/18. 蔵BAR秦内 (content, sabae)	8/6. つづじバスマップ (map, sabae, opengovjp, ydn)	9/8. 河和田アートキャンプ (map, sabae, opengovjp, ydn)	9/10. 鮎江地域活性化プランコン テスト写真 (content, map, sabae)	9/12. 駐車場マップ (tool, sabae, map)	9/18. 鮎江文化財タイル表示 (content, sabae)	10/6. つづじバス路線テスト (sabae, map)

2012年
鮎江のアプリ
100公開

<https://fukuno.jig.jp/2012/#sabae>



まだ緯度経度で消耗してるの？

緯度経度のここがイケてない

緯度 35.658633790016204, 経度 139.74546563023935

1. 2つの使わないといけない（どっちが緯度？）
2. 精度を表す方法が別に必要
3. 浮動小数表現にすると微妙に変わってしまう

Geo3x3

ジオスリー・バイスリー

Geo3x3 (ジオスリー・バイスリー)

E (East 東経)

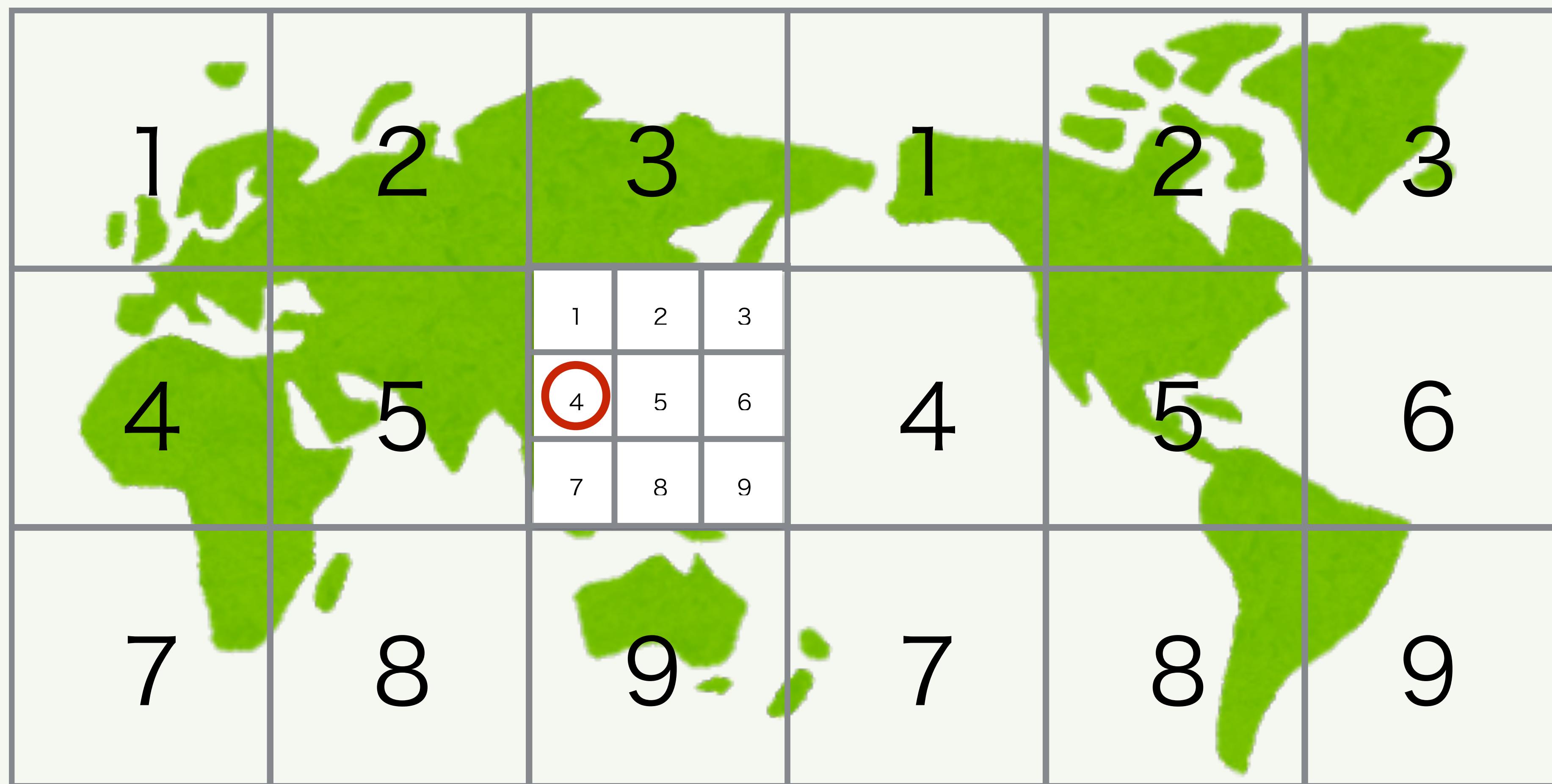
W (West 西経)



Geo3x3: E64

E (East 東経)

W (West 西経)

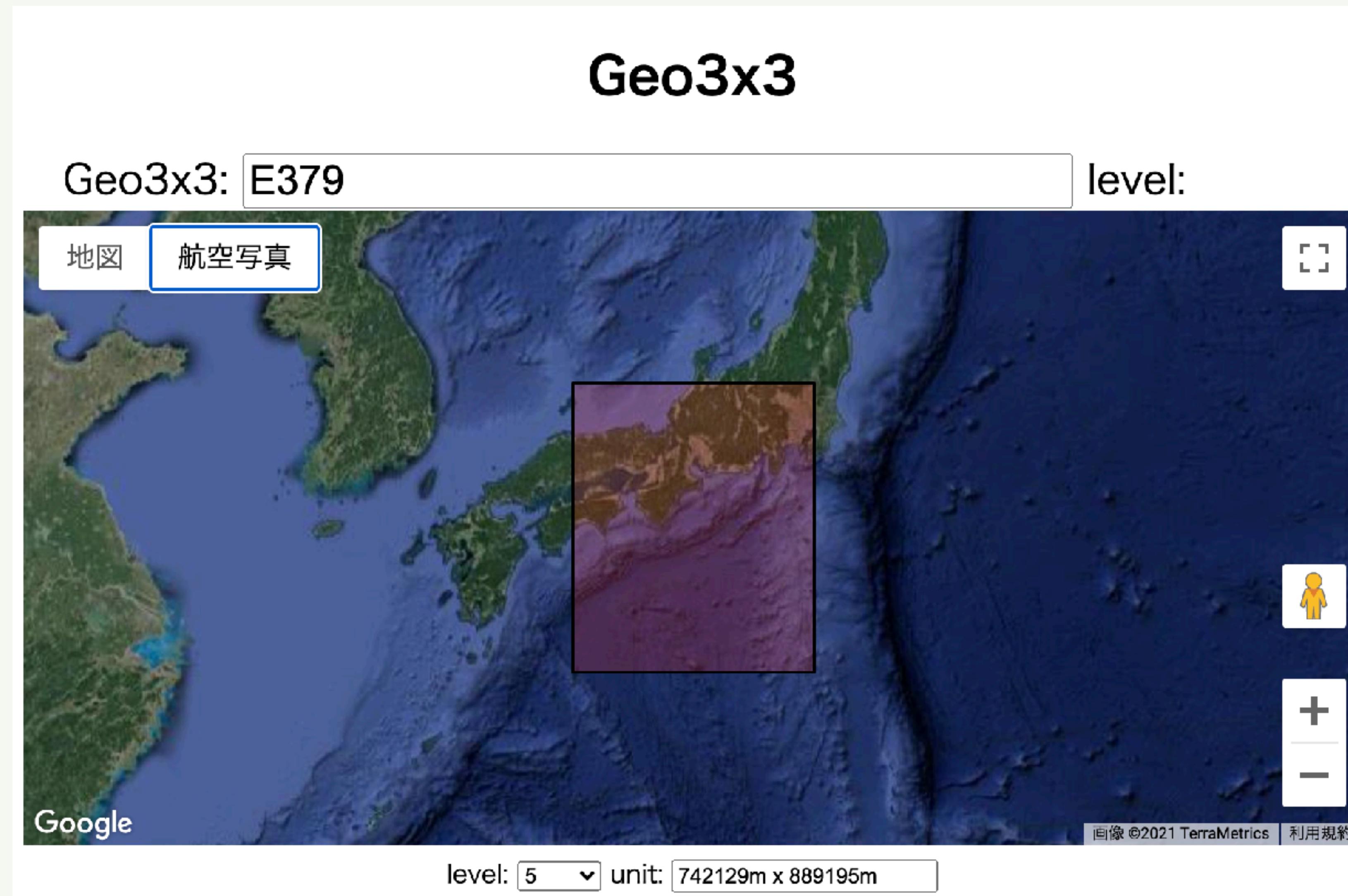


Geo3x3: E3793653391822



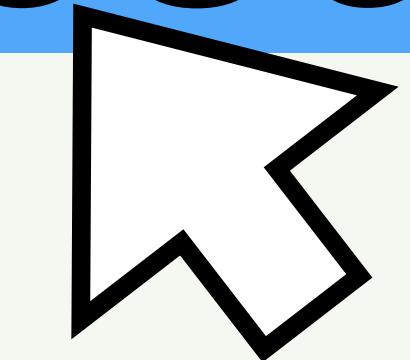
東京タワー

位置をざっくり表せる



コピペビリティが高い！

E379365391822



だぶるくりっく！Ctrl+C

34言語サポート！

34 programming languages supported now

現在34のプログラミング言語対応しています

([JavaScript](#) / [TypeScript](#) / [C](#) / [C++](#) / [C#](#) / [Swift](#) / [Java](#) / [Python](#) / [Ruby](#) / [PHP](#) / [Go](#) / [Kotlin](#) / [Dart](#) / [Rust](#) / [Haskell](#) / [OpenVBS](#) / [Scala](#) / [R](#) / [GAS](#) / [Nim](#) / [Lua](#) / [Perl](#) / [Elixir](#) / [Groovy](#) / [D](#) / [Julia](#) / [Racket](#) / [OCaml](#) / [Erlang](#) / [Clojure](#) / [F#](#) / [Haxe](#) / [Scheme](#) / [Common Lisp](#))



Cyber Valley, Japan: E379219

Geo.d	add D
Geo3x3.R	add R
Geo3x3.cs	add PHP / TypeScript
Geo3x3.d	add D
Geo3x3.dart	improve samples
Geo3x3.groovy	improve samples
Geo3x3.gs	add GAS
Geo3x3.hs	fail → error
Geo3x3.hx	add Haxe
Geo3x3.java	add PHP / TypeScript
Geo3x3.jl	improve julia
Geo3x3.js	fix err on Safari
Geo3x3.kt	improve samples
Geo3x3.mjs	fix err on Safari
Geo3x3.obs	add OpenVBS
Geo3x3.php	add PHP / TypeScript
Geo3x3.r.c	[fix] decode W bug
Geo3x3.r.h	[add] C (recursive style)
Geo3x3.scala	fix indent

今、一番好きな言語
JavaScript (ES6)

import可能！

```
<script type="module">  
import { CSV } from "./CSV.js";  
</script>
```

URLで指定可能！

```
<script type="module">  
import { CSV } from "https://  
code4sabae.github.io/js/CSV.js";  
</script>
```

Denoを使えばそのままサーバーで動く！

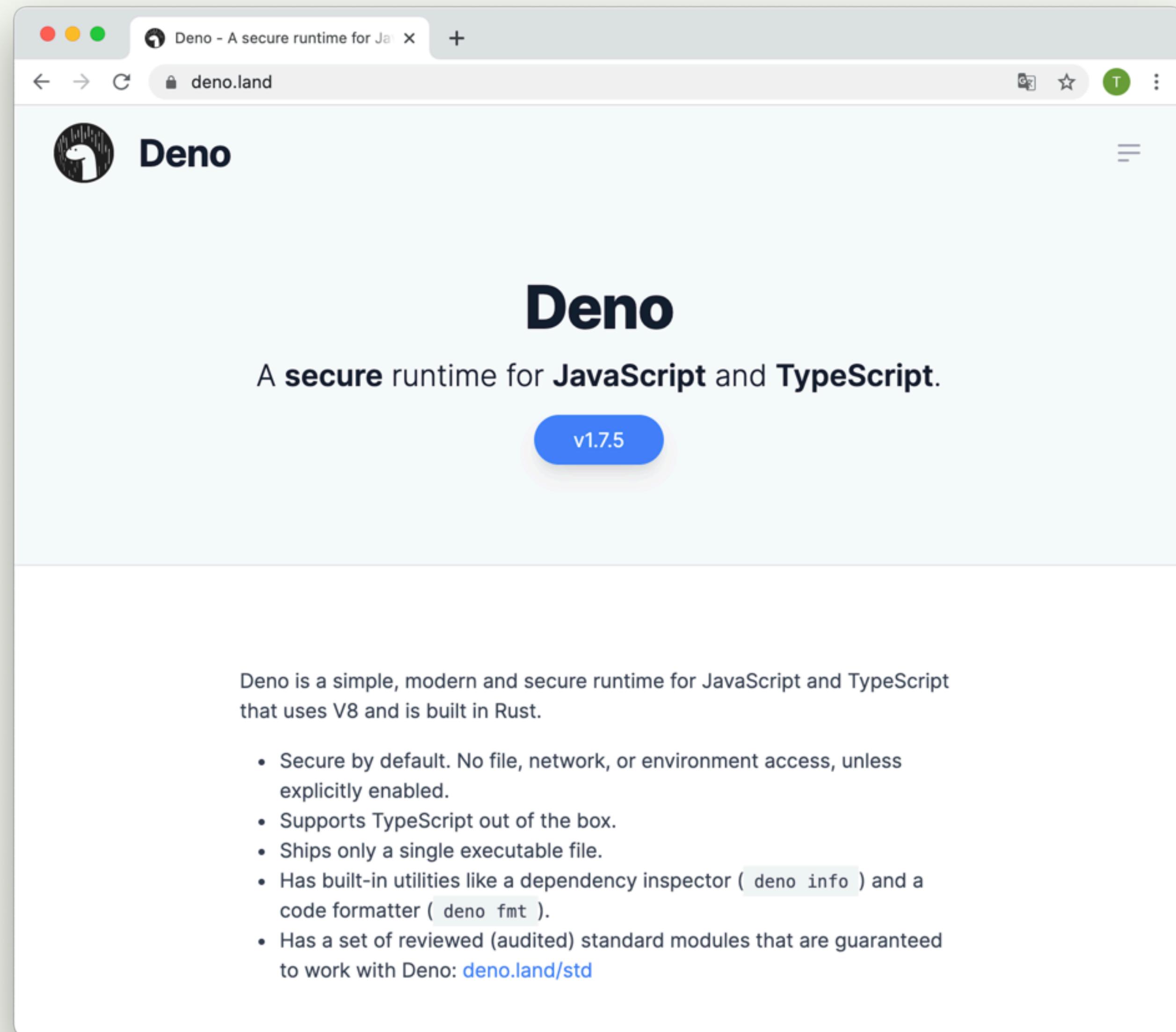
```
import { CSV } from "https://  
code4sabae.github.io/js/CSV.js";
```

```
$ deno run test.js
```

フロントもバックエンドも
超入門

1. Denoをインストール
2. サーバーを書く(API)
3. フロントを書く(DOM)

1. Denoをインストール
2. サーバーを書く(API)
3. フロントを書く(DOM)



<https://deno.land/>

コンソール（コマンドプロンプトかターミナル）

```
$ deno
```

```
> 1+1  
2
```

```
> 1 << prompt()  
Prompt: 16  
65536
```

止めるときは Ctrl+D

1. Denoをインストール
2. サーバーを書く(API)
3. フロントを書く(DOM)

適当に testserver とかでディレクトリ作成

```
import { Server } from "https://js.sabae.cc/Server.js"  
new Server(8001);
```

server.js で保存

コンソール（コマンドプロンプトかターミナル）

```
$ deno run -A server.js
```

http://localhost:8001 をブラウザで開く

testserver に static ディレクトリを作成し、index.html をつくる

```
<!DOCTYPE html><html><head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width">
</head><body>
  Hello!
</body></html>
```

<http://localhost:8001> をブラウザで開く、見える！レスポンシブ！

server.jsを編集

```
import { Server } from "https://js.sabae.cc/Server.js"
class MyServer extends Server {
    api(path) {
        return path.toUpperCase();
    }
}
new MyServer(8001);
```

コンソール、一度、Ctrl+C で止めて、上押して、エンター

```
$ deno run -A server.js
```

http://localhost:8001/api/test とかをブラウザで開く

server.jsを編集

```
import { Server } from "https://js.sabae.cc/Server.js"
const list = [];
class MyServer extends Server {
    api(path) {
        list.push(path);
        return list;
    }
}
new MyServer(8001);
```

server.jsを編集（オブジェクトを作つて返す）

```
import { Server } from "https://js.sabae.cc/Server.js"
class MyServer extends Server {
    api(path) {
        return { name: "jigintern", path: path };
    }
}
new MyServer(8001);
```

http://localhost:8001/api/test とかをブラウザで開く

```
{"name":"jigintern","path":"/api/test"}
```

↑ JSON です

1. Denoをインストール
2. サーバーを書く(API)
3. フロントを書く(DOM)

testserver に static/index.html を編集

```
</head><body>  
<script type=module>  
alert("Hi! " + prompt());  
</script>  
</body></html>
```

<http://localhost:8001> をブラウザで開く、動く！挨拶してくれるよ

testserver に static/index.html を編集

```
</head><body>  
<script type=module>  
import { fetchJSON } from "https://js.sabae.cc/fetchJSON.js";  
window.onload = async () => {  
    const data = await fetchJSON("api/test");  
    alert(data.name);  
};  
</script>  
</body></html>
```

※ `async() =>` は、`await`を使う引数なしの関数を作る宣言

testserver に static/index.html を編集

```
</head><body>  
<script type=module>  
import { fetchJSON } from "https://js.sabae.cc/fetchJSON.js";  
window.onload = async () => {  
    const data = await fetchJSON("api/test", { name: "jig", age: 17});  
    alert(data.name);  
};  
</script>  
</body></html>
```

※ fetchJSON関数を使ってパラメータをサーバーに渡す

testserver に static/index.html を編集

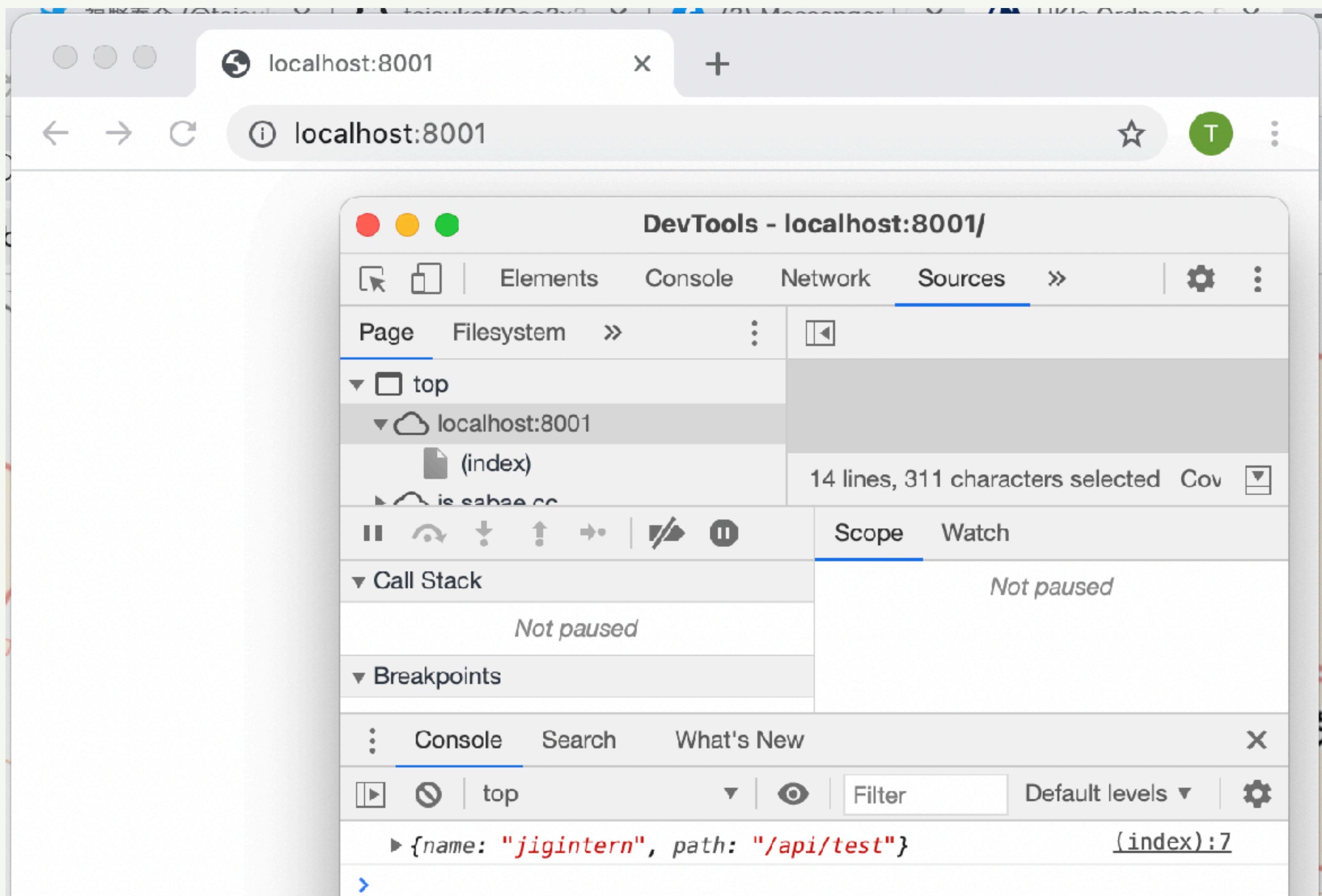
```
</head><body>  
<div id=name></div>  
<script type=module>  
import { fetchJSON } from "https://js.sabae.cc/fetchJSON.js";  
window.onload = async () => {  
    const data = await fetchJSON("api/test", { name: "jig", age: 17 });  
    name.textContent = data.name;  
};  
</script>  
</body></html>
```

※divタグにnameというidをつけて、受け取ったデータをセット (DOM操作)

testserver に static/index.html を編集

```
</head><body>  
<script type=module>  
import { fetchJSON } from "https://js.sabae.cc/fetchJSON.js";  
window.onload = async () => {  
    const data = await fetchJSON("api/test", { name: "jig", age: 17 });  
    alert(data.name);  
    console.log(data);  
};  
</script>  
</body></html>
```

※ サーバーでもフロントでも、デバッグに便利な `console.log`



開発ツールで
見てみよう！

Mac: Cmd + Opt + I
Windows: F12

チーム開発 with GitHub

チームで使うGitHubの基本

1. ブランチ切る（別バージョン作る）
2. 編集する
3. コミットする
4. プルリク（更新依頼）送る
5. 誰かが確認して、マージ（更新）する



Search or jump to...



Pull requests Issues Marketplace Explore



jigintern

Repositories 17

Packages

People 45

Teams 15

Find a repository...

Type: All ▾

Language: All ▾

New

[2021_winter_e](#)

0 0 0 0 0 Updated 6 days ago

Top languages

JavaScript Vue Go C#
HTML

[2021_winter_d](#)

0 0 0 0 0 Updated 6 days ago

People

45 >

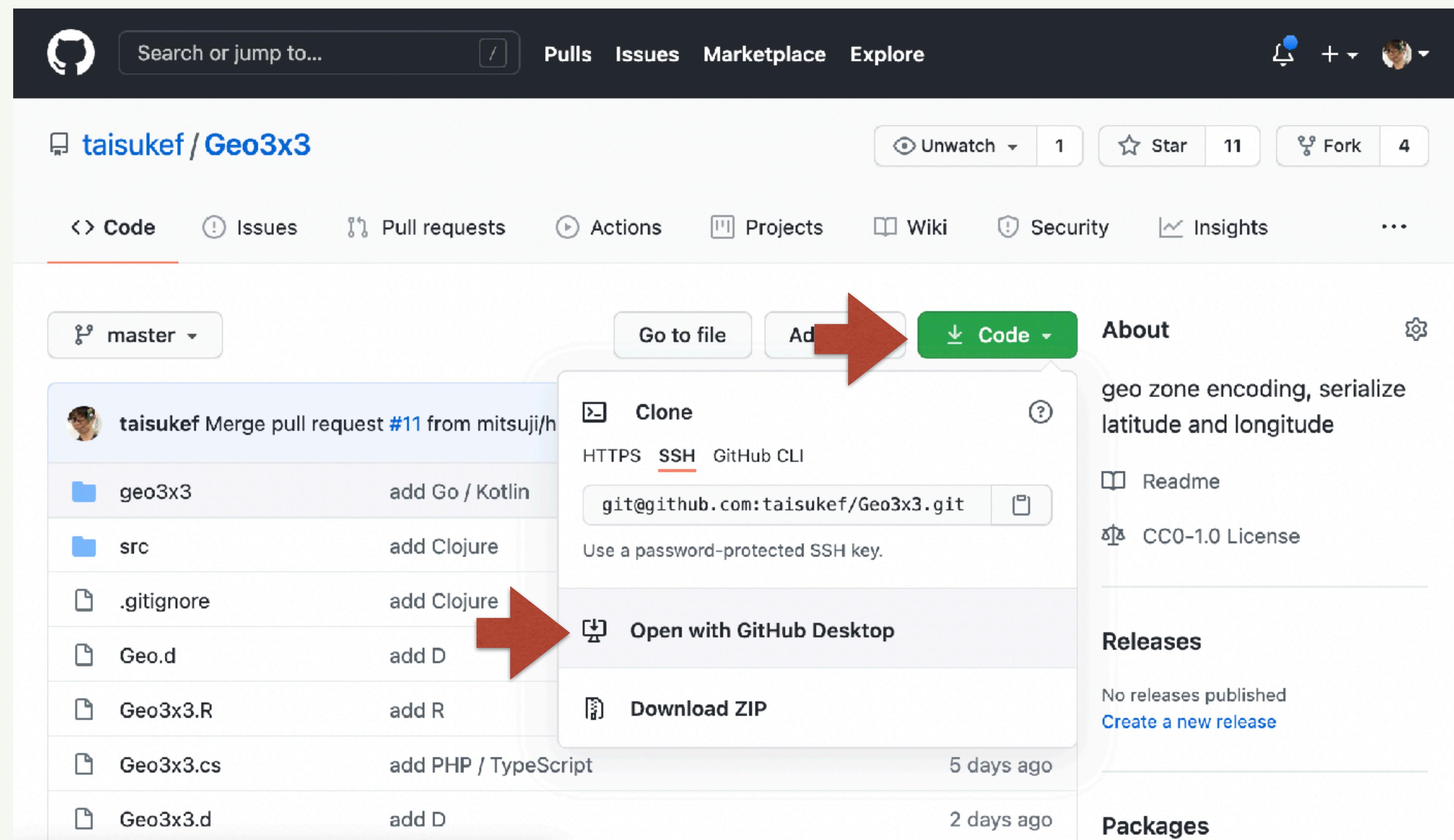


[2021_winter_c](#)

0 0 0 0 0 Updated 6 days ago

<https://github.com/jigintern/>

GitHub Desktop で開く



3. Push origin で GitHubを更新

1. 更新したものが表示されている
確認

2. 変更点を
わかるように記述

The screenshot shows a GitHub desktop application window. At the top, it displays the repository name 'Current Repository js' and the branch 'Current Branch master'. Below this, the 'Changes' tab is selected, showing '1 changed file' named 'fetchJSON.js'. The main area displays a diff view of the file's contents. The diff highlights changes made to the code, specifically changing the variable 'opt' from being defined in the function parameters to being defined as a local variable within the function. The bottom part of the window shows a commit message input field containing 'Update fetchJSON.js' and a 'Commit to master' button.

```
@@ -1,11 +1,11 @@
1   1 const fetchJSON = async (url, req) => {
2   2 - const opt = {
3   3 + const opt = req ? {
4       method: "POST",
5       mode: "cors",
6       cache: "no-cache",
7       headers: { "Content-Type": "application/json" },
8       body: JSON.stringify(req),
9     };
10  10 + } : null;
11  11 const res = await (await fetch(url, opt)).json();
12  12 return res;
13  13};
```