

サイバー大学IT総合学部

専門応用科目

JavaScriptフレームワークによるWebプログラミング

# 第8回 データベース接続（続き）、Web API連携

小園井康志

## 第8回 学習目標

- Node.js ExpressからMongoDBへの接続プログラムの作成ができる
- ネットで公開されているWeb APIについて例をあげ概要を説明することができる
- 公開されたWeb APIにアクセスすることができる
- そのWeb APIにNode.js Expressからアクセスするプログラムの作成ができる

# 第8回 授業構成

- 第1章 DB連携プログラムの作成
- 第2章 Web APIについて
- 第3章 Web APIにアクセスしてみる
- 第4章 Node.jsからWeb APIの利用

JavaScriptフレームワークによるWebプログラミング  
第8回

# 第1章

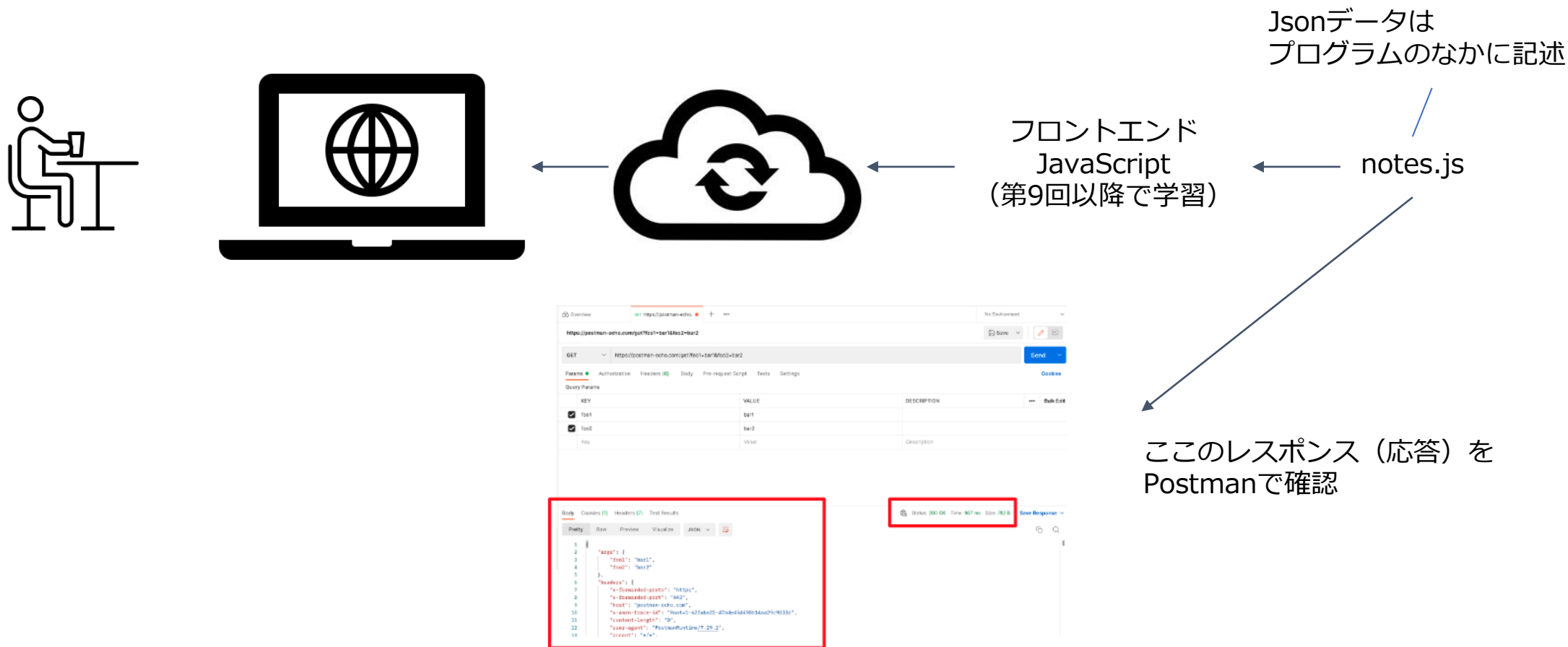
## DB連携プログラムの作成

# 第1章 学習目標

- Node.js ExpressからMongoDBへの  
接続プログラムの作成ができる

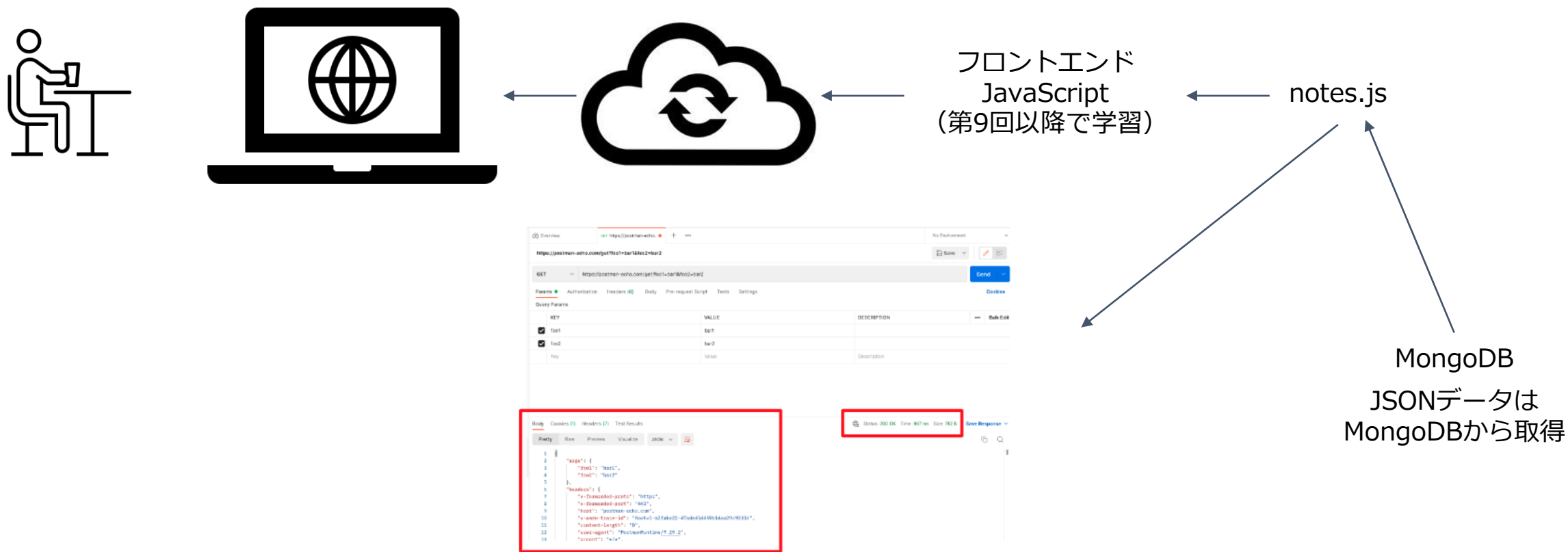
# REST APIを利用した画面作成

第6回では以下のようにサービスを作成



# MongoDBとの連携

## 今回はMongoDBのデータをRest APIで応答するサービスを作成



# プログラムの場所について

今回のプログラムは  
前回、第6回まで作成した場所（myapp）に作成

（ご自分のホームディレクトリなど） - myapp - （今回のプログラム）  
|  
-- express\_mongodb



# notes.jsの変更

## routes配下のnotes.jsを修正

```
var express = require('express');
var router = express.Router();

// 接続情報を設定
const { MongoClient } = require("mongodb");
const uri = "mongodb+srv://koh:kohtaroh01xxx@cluster0.y5uxhta.mongodb.net/food?retryWrites=true&w=majority ";
const client = new MongoClient(uri);

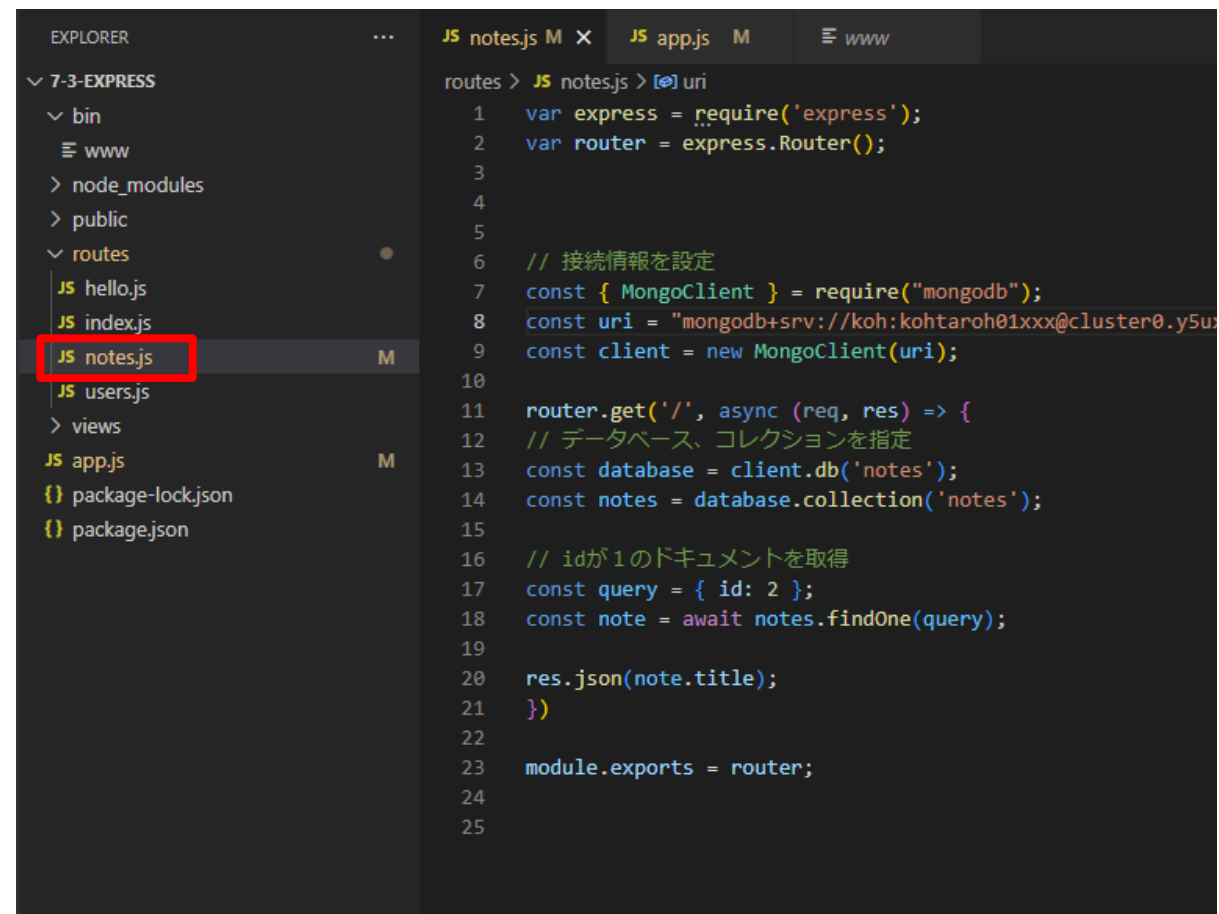
router.get('/', async (req, res) => {
  // データベース、コレクションを指定
  const database = client.db('notes');
  const notes = database.collection('notes');

  // idが1のドキュメントを取得
  const query = { id: 2 };
  const note = await notes.findOne(query);

  res.json(note);
})

module.exports = router;
```

ここは自分の値に変更



The screenshot shows the VS Code interface. On the left, the 'EXPLORER' sidebar displays the project structure for '7-3-EXPRESS'. The 'routes' directory is expanded, and 'notes.js' is highlighted with a red rectangle. On the right, the editor shows the code for 'notes.js'. The code is identical to the one shown in the first block, but with a few differences: line 8 has a truncated URI, and line 12 has a comment in Japanese. The line numbers 1 through 25 are visible on the left side of the editor.

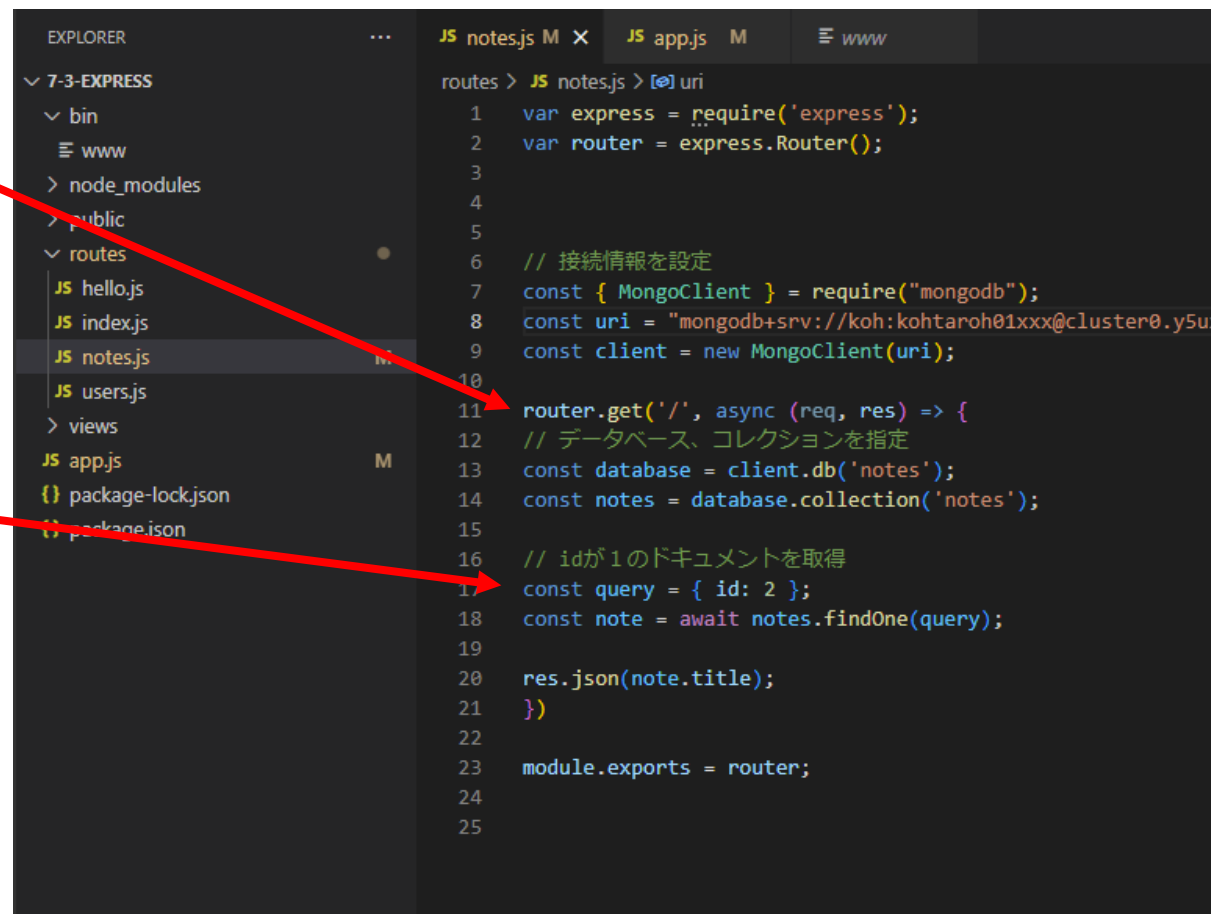
```
JS notes.js M X JS app.js M www
7-3-EXPRESS
  bin
  www
  node_modules
  public
  routes
    JS hello.js
    JS index.js
    JS notes.js M
    JS users.js
  views
  JS app.js M
  package-lock.json
  package.json

routes > JS notes.js > uri
1 var express = require('express');
2 var router = express.Router();
3
4
5
6 // 接続情報を設定
7 const { MongoClient } = require("mongodb");
8 const uri = "mongodb+srv://koh:kohtaroh01xxx@cluster0.y5uxhta.mongodb.net/food?retryWrites=true&w=majority ";
9 const client = new MongoClient(uri);
10
11 router.get('/', async (req, res) => {
12   // データベース、コレクションを指定
13   const database = client.db('notes');
14   const notes = database.collection('notes');
15
16   // idが1のドキュメントを取得
17   const query = { id: 2 };
18   const note = await notes.findOne(query);
19
20   res.json(note.title);
21 })
22
23 module.exports = router;
24
25
```

# notes.jsの中身

データベースの指定

データベースからデータを取得



```
EXPLORER
7-3-EXPRESS
  bin
  www
  node_modules
  public
  routes
    hello.js
    index.js
    notes.js
    users.js
  views
  app.js
  package-lock.json
  package.json


JS notes.js M X JS app.js M www
routes > JS notes.js > uri
1  var express = require('express');
2  var router = express.Router();
3
4
5
6  // 接続情報を設定
7  const { MongoClient } = require("mongodb");
8  const uri = "mongodb+srv://koh:kohtaroh01xxx@cluster0.y5u";
9  const client = new MongoClient(uri);
10
11 router.get('/', async (req, res) => {
12   // データベース、コレクションを指定
13   const database = client.db('notes');
14   const notes = database.collection('notes');
15
16   // idが1のドキュメントを取得
17   const query = { id: 2 };
18   const note = await notes.findOne(query);
19
20   res.json(note.title);
21 })
22
23 module.exports = router;
24
25
```

# app.jsの確認

var notesRouter =  
require('./routes/notes')

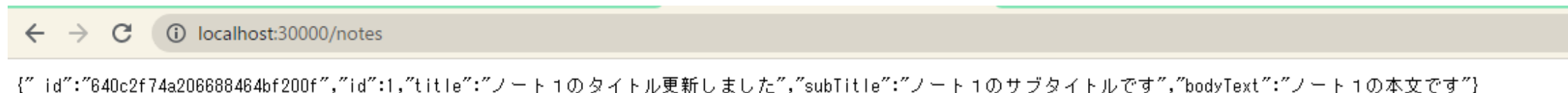
app.use('/notes', notesRouter);

```
7  var indexRouter = require('./routes/index');
8  var usersRouter = require('./routes/users');
9  var helloRouter = require('./routes/hello');
10 var notesRouter = require('./routes/notes');
11
12 var app = express();
13
14 // view engine setup
15 app.set('views', path.join(__dirname, 'views'));
16 app.set('view engine', 'jade');
17
18 app.use(logger('dev'));
19 app.use(express.json());
20 app.use(express.urlencoded({ extended: false }));
21 app.use(cookieParser());
22 app.use(express.static(path.join(__dirname, 'public')));
23
24 app.use('/', indexRouter);
25 app.use('/users', usersRouter);
26 app.use('/hello', helloRouter);
27 app.use('/notes', notesRouter);
28
```



# ブラウザでの動作確認

- コマンドで以下を入力  
`% npm install mongodb`
- npm startでアプリを起動し、ブラウザに、以下を入力。  
<http://localhost:30000/notes>
- 以下のように表示されたらOK。  
(\*ポート番号は自分のものを指定)



もし結果がnullになっている場合データベースの中身が空であることが考えられます。  
第7回の3章の演習、insertMany.jsを実行してみてください

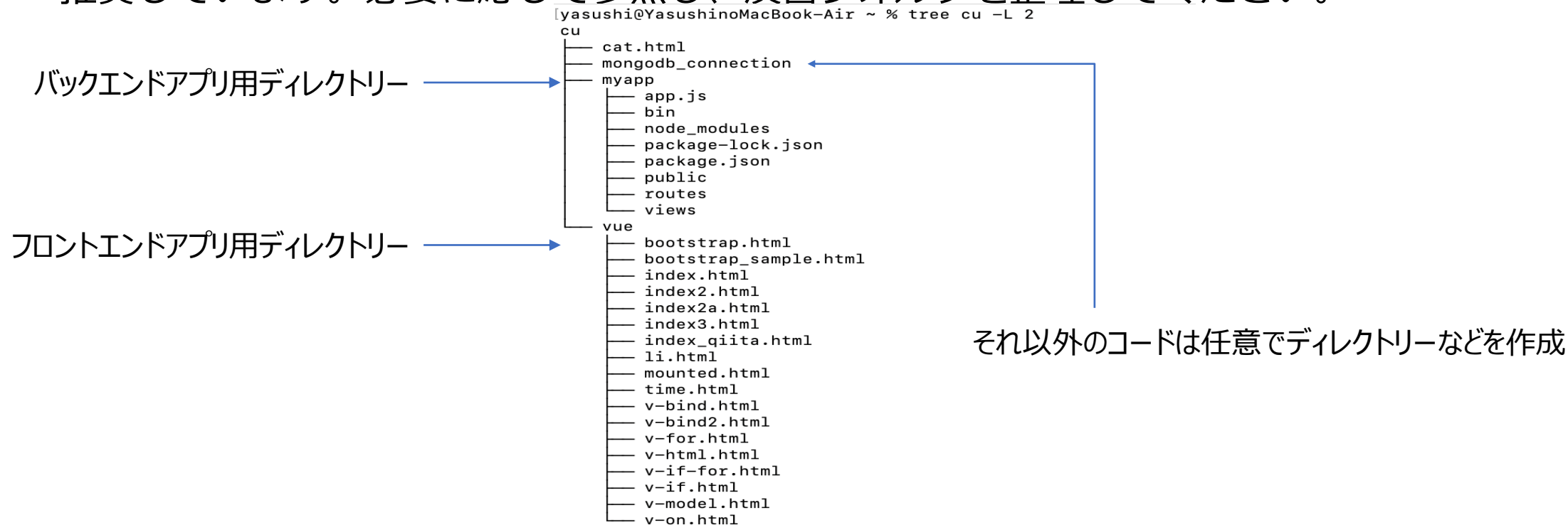
# 実機演習：MongoDBとの連携～ブラウザでの動作確認

動画を全画面で視聴してください

# 実習時間

- ・ 10分程度を目安に動画を止めて前ページまでの実習をしてください。
- ・ 作業が終わったらビデオを再開して学習を進めてください。

※第1回4章で説明した通り、実習のファイル作成時には以下のディレクトリ構造を推奨しています。必要に応じて参照し、演習フォルダを整理してください。



# サーバでの動作確認

- GitHubへのコードのプッシュ
- サーバへのログイン
- Git Pullでサーバのコードを最新に
- サーバでアプリ起動
- Postmanで動作確認

アプリの確認のためのURLは  
(自分のサーバのIPアドレス:ポート番号/notes)  
例: 153.120.121.157:30000/notes

\* 接続情報はGitHubにそのままの形ではあげない  
GitHubにあげると世界中の人に見られてしまう可能性がある  
一度違う文字に変換か削除してサーバ上で編集、追加

# 実機演習：サーバでの動作確認

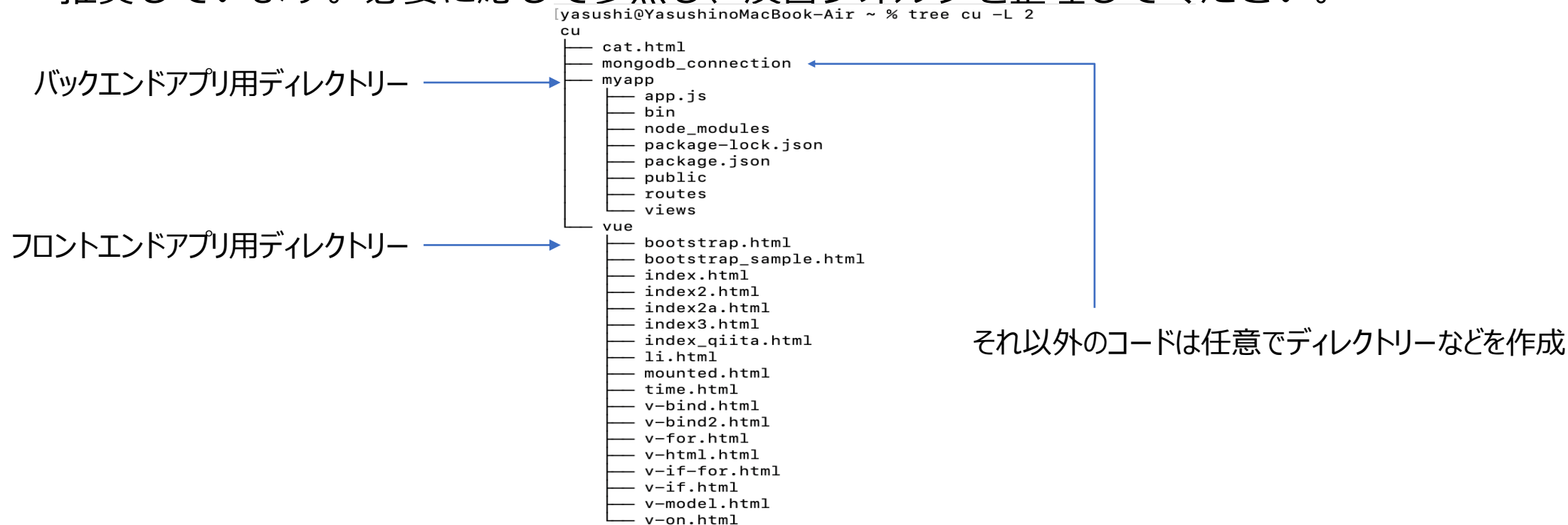
動画を全画面で視聴してください



# 実習時間

- ・ 10分程度を目安に動画を止めて前ページまでの実習をしてください。
- ・ 作業が終わったらビデオを再開して学習を進めてください。

※第1回4章で説明した通り、実習のファイル作成時には以下のディレクトリ構造を推奨しています。必要に応じて参照し、演習フォルダを整理してください。



# サーバでの動作確認

- サーバでのファイル編集にはサーバ上のエディタ、vimを使用  
vim:Linux、Mac上のコマンドラインで動作するテキストエディタ

- 使い方

% vim ファイル名

最初に起動したときは閲覧モードなのでキーボードの **i** をおして編集モード（insert）にして編集する。

編集が終わったら **Esc** キーをおして **:** コロンに続き **w**、**q** とキー入力。**w** は保存、**q** は終了。

- その他のコマンド

編集結果を保存しないで終了 **q**、**!**

# 第1章 まとめ

- Node.jsとMongoDBの連携プログラムを作成、動作確認を行った
  - 前回までのプログラムを修正してMongodbのデータを取得するように変更
  - ローカル環境で動作確認
  - サーバ環境での動作をPostmanで確認

JavaScriptフレームワークによるWeb開発  
第8回

# 第1章 DB連携プログラムの作成

終わり

JavaScriptフレームワークによるWebプログラミング  
第8回 Web API連携

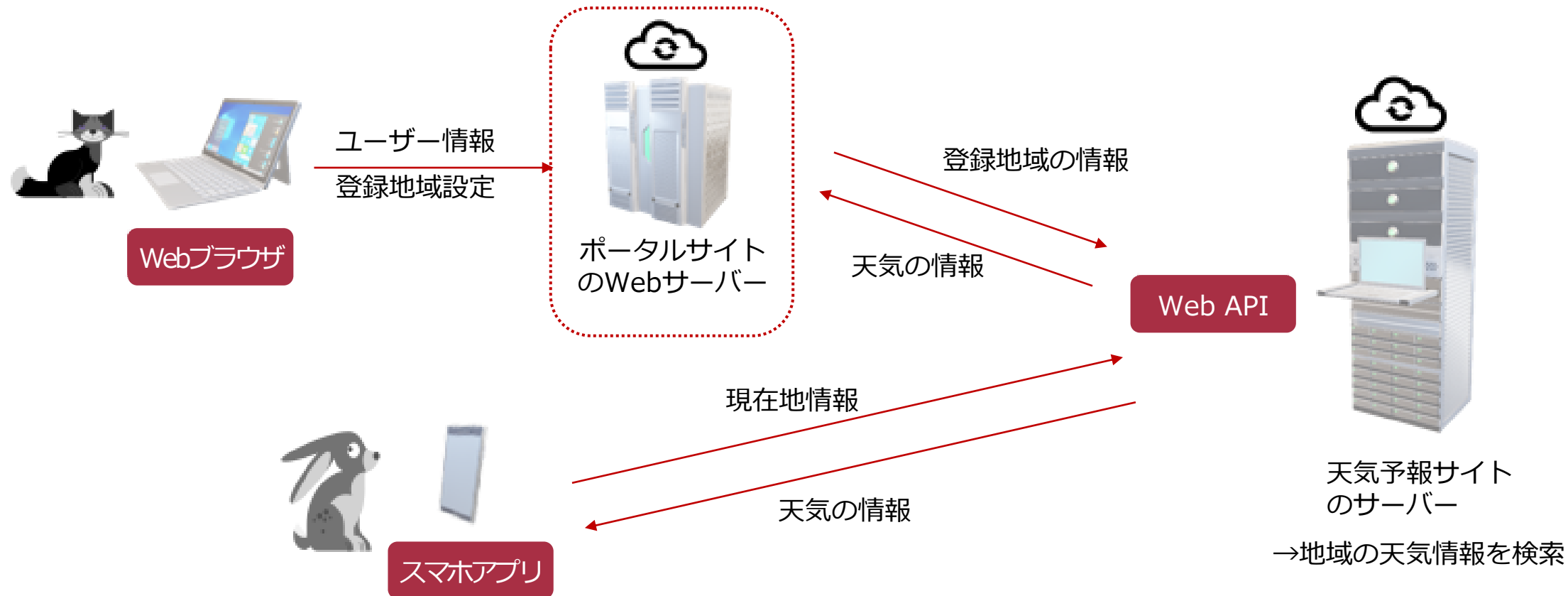
## 第2章

# Web APIについて

## 第2章 学習目標

- 代表的なWeb APIサービスの概要を理解し説明できる
- Web APIの認証方法について理解し説明できる

# Web API使用例（おさらい）



# 世の中に公開されているWeb API例

以下に、現在公開されているWeb APIをいくつか紹介する。  
事前に開発者キーの申請や各種登録が必要なものもある。

## OpenWeatherMapAPI



天気情報を提供するAPIで、世界中の都市の天気や気温、風速などが取得できる

<https://openweathermap.org/>

## NASA API



NASAが提供するAPIで、宇宙に関するデータや画像、地球の衛星写真などが取得できる

<https://data.nasa.gov/>



# 世の中に公開されているWeb API例

以下に、現在公開されているWeb APIをいくつか紹介する。  
事前に開発者キーの申請や各種登録が必要なものもある。

## Giphy API



GIFアニメーションを提供するAPIで、  
キーワードに応じて様々なGIFを取得できる  
<https://developers.giphy.com/>

## The New York Times API

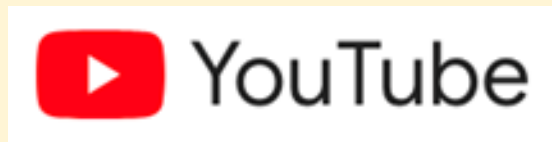


ニューヨーク・タイムズが提供するAPIで、  
ニュース記事や写真、ブログなどを取得できる  
<https://developer.nytimes.com/docs/articlesearch-product/1/overview>

# 世の中に公開されているWeb API例

以下に、現在公開されているWeb APIをいくつか紹介する。  
事前に開発者キーの申請や各種登録が必要なものもある。

## YouTube Data API



YouTubeのデータを提供するAPIで、動画の検索や再生、チャンネル情報などが取得できる

<https://developers.google.com/youtube/v3/getting-started?hl=ja>

## Advice Slip API

ADVICE SLIP JSON API

アドバイスや格言を提供するAPIで、簡単なHTTPリクエストで取得できる

<https://api.adviceslip.com/>

# 世の中に公開されているWeb API例

以下に、現在公開されているWeb APIをいくつか紹介する。  
事前に開発者キーの申請や各種登録が必要なものもある。

## NewsAPI



世界中のニュース記事を提供するAPIを提供。  
News APIを使用して、アプリケーションや  
ウェブサイトにニュース記事を表示することができる  
<https://newsapi.org/s/japan-news-api>

## Pixabay API



高品質な無料のストック写真やビデオを提供するサイト。  
Pixabayの写真やビデオをアプリケーションや  
ウェブサイトに表示することができる  
<https://pixabay.com/ja/service/about/api/>

# 動画説明：世の中に公開されているWeb API例

動画を全画面で視聴してください

# Web APIと認証

- サービスを利用する際にそのサービスにアクセスできることを、許可されているかどうかを確認する

## APIキー認証

- 非常にシンプルに認証を実現することができる
- APIキーと呼ばれる文字列で認証を行う

## アクセストークン認証

- ログインID・パスワードなどでユーザー認証を行なった後に、サービスから発行されるアクセストークンを受け取って、APIのリクエスト時に送信する方式
- より高いセキュリティを保つことができる

※今回の実習では使わない

## 第2章 まとめ

- 第7回までに学習したREST APIの他にも、世の中には無料で使用可能なWeb APIが存在しており、その概要を学習した。
- Web APIの認証制度について学習した。
  - APIキー認証
  - アクセスキー認証

JavaScriptフレームワークによるWebプログラミング  
第8回 Web API連携

# 第2章

# Web APIについて

## 終わり

JavaScriptフレームワークによるWebプログラミング  
第8回 Web API連携

## 第3章

# Web APIにアクセスしてみる



## 第3章 学習目標

- 実際にWeb APIにアクセスし、使用することができる

# The Cat APIを使ってみる

ここでは、The Cat APIを使用して、猫の画像を取得してみます。

- <https://thecatapi.com/>
- ランダムに可愛い猫の画像を提供するAPI。  
curlなどで簡単にアクセス可能



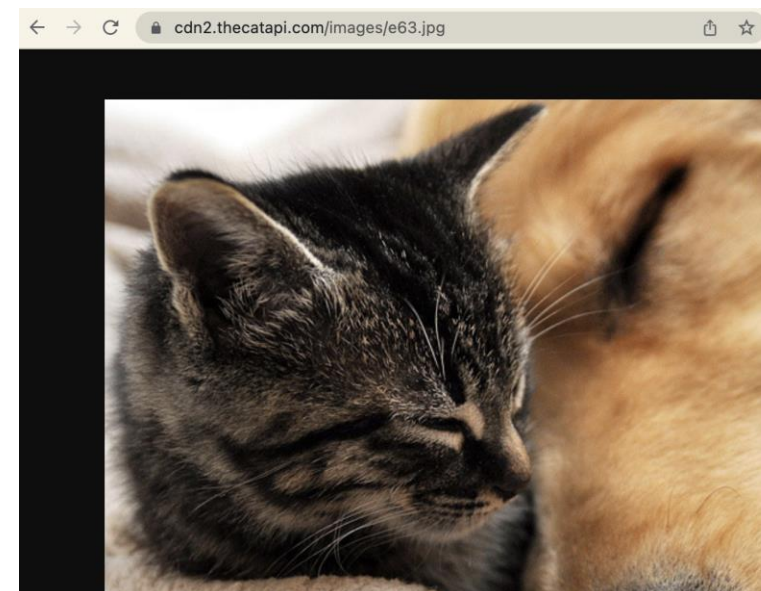
# The Cat APIを使ってみる

ブラウザを起動して、次のURLを入力する。

- <https://api.thecatapi.com/v1/images/search>
- ブラウザによって応答の表示形式が変わりますが、毎回URLの値が変化するので、そのURL（下図の赤枠部分）をコピーし、ブラウザでアクセスすると、猫の画像が表示される。

← → ↻ [api.thecatapi.com/v1/images/search](https://api.thecatapi.com/v1/images/search)

[{"id":"MTkyMTY1OQ","url":"<https://cdn2.thecatapi.com/images/MTkyMTY1OQ.jpg>", "width":1024,"height":768}]



# The Cat APIを使ってみる

ブラウザでfirefoxを使っている場合

Firefoxの場合

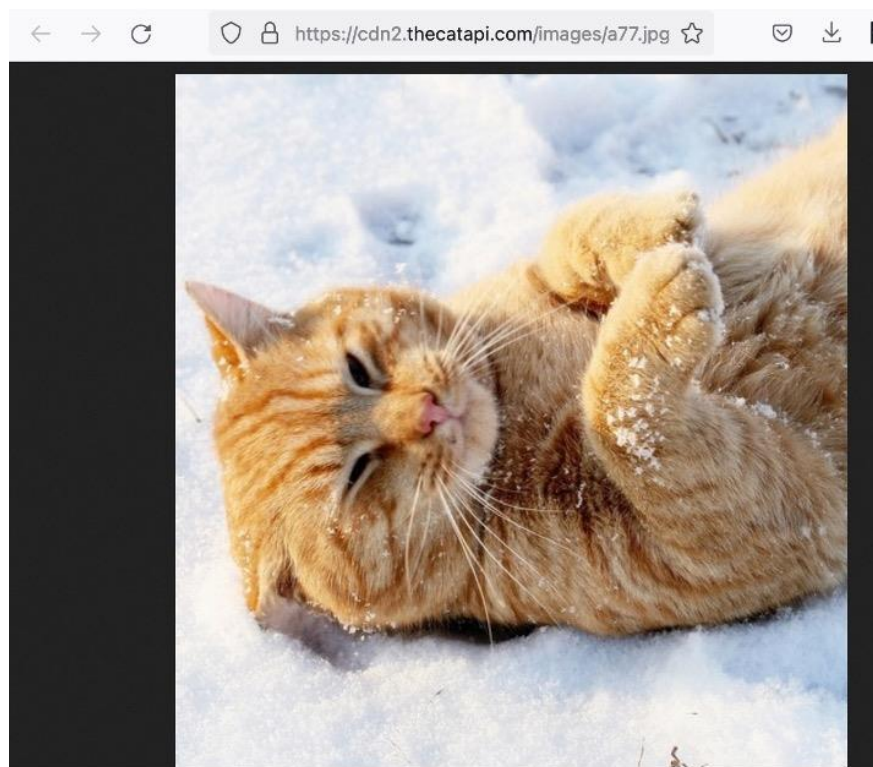


Firefoxの場合は、赤枠部分をクリックすると画像が表示される

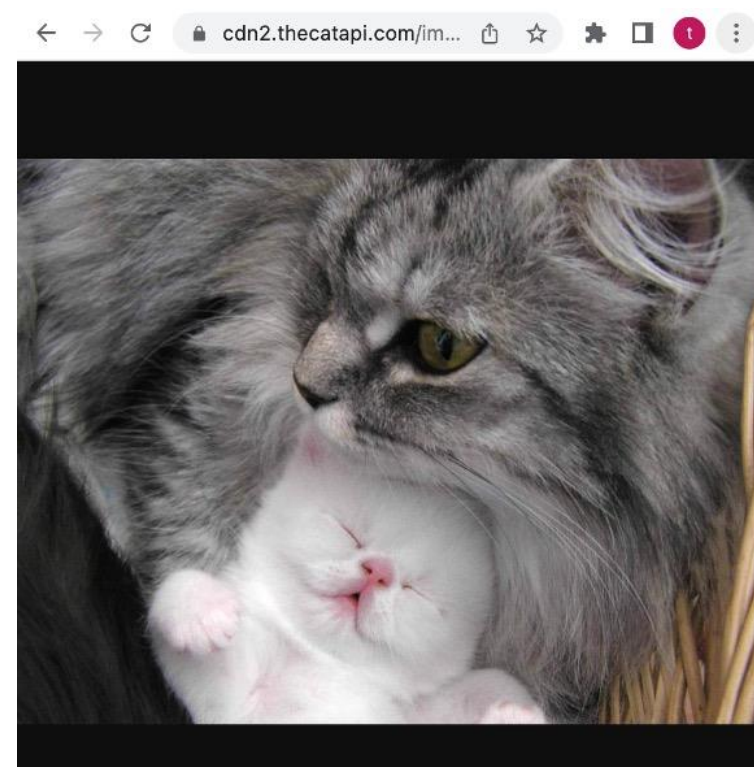
# The Cat APIを使ってみる

前ページで表示されたURLをブラウザで表示すると  
猫の画像が表示される

## Firefoxの場合



## Chromeの場合

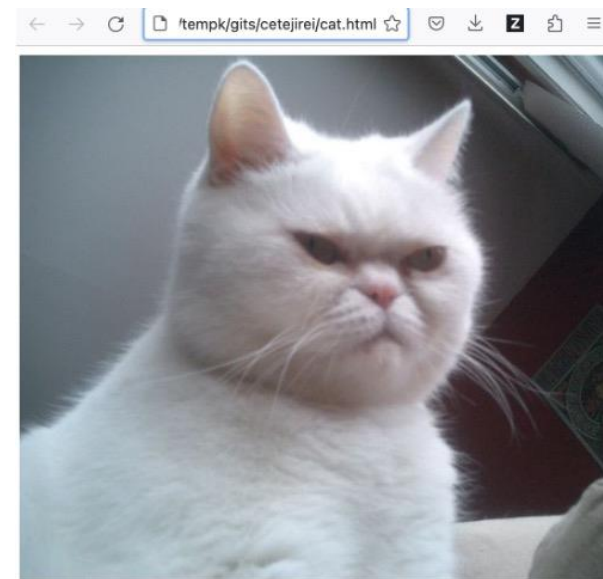


# The Cat APIを使ってみる

The Cat APIを呼び出し、結果のURLの猫の画像をブラウザに表示するhtmlファイルを作成し、実行する。

- 以下のコードをコピーし、cat.htmlというファイル名で保存する。
- 保存したcat.htmlをブラウザで開くと猫の画像が表示される。

```
<!DOCTYPE html>
<html>
  <head>
    <title>Cat API Example</title>
  </head>
  <body>
    <img id="cat-image" src="" alt="Random cat image">
    <script>
      const request = new XMLHttpRequest();
      request.open('GET', 'https://api.thecatapi.com/v1/images/search');
      request.onload = function () {
        const data = JSON.parse(request.responseText);
        const catImage = document.getElementById('cat-image');
        catImage.src = data[0].url;
      };
      request.send();
    </script>
  </body>
</html>
```



cat.htmlをWebブラウザで開くと猫の画像が表示される

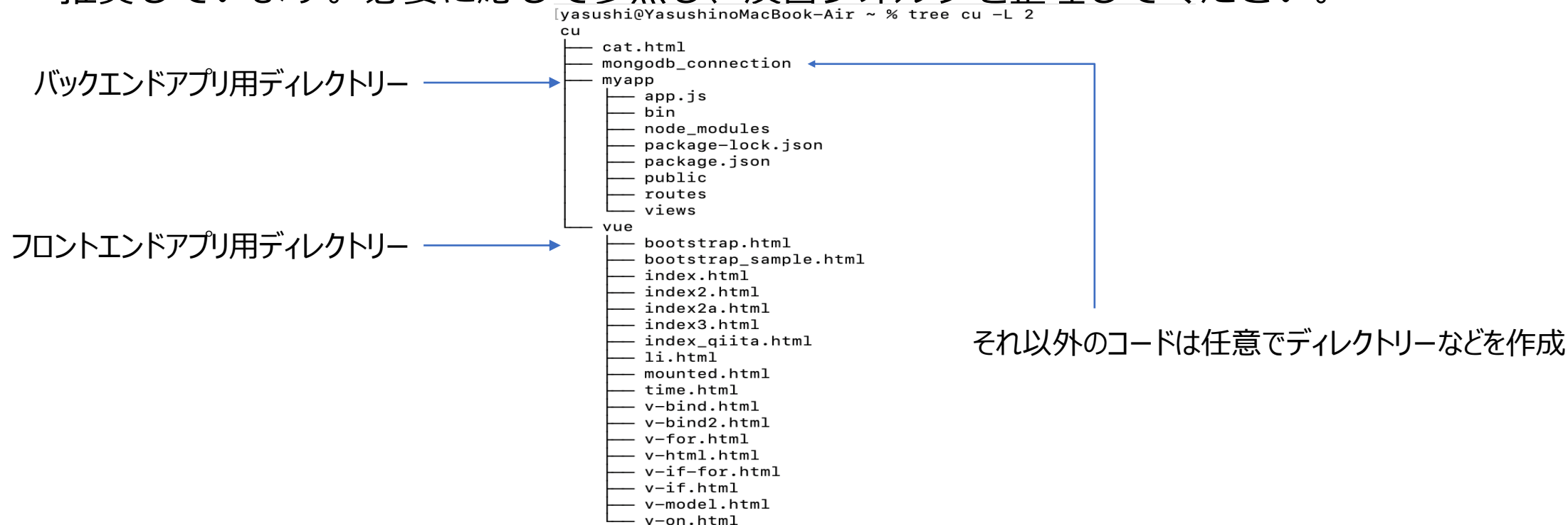
# 実機演習：The Cat APIを使ってみる

動画を全画面で視聴してください

# 実習時間

- ・ 10分程度を目安に動画を止めて前ページまでの実習をしてください。
- ・ 作業が終わったらビデオを再開して学習を進めてください。

※第1回4章で説明した通り、実習のファイル作成時には以下のディレクトリ構造を推奨しています。必要に応じて参照し、演習フォルダを整理してください。





# The Cat APIを使ってみる

## プログラム内容

今回はnode.jsのrequestモジュールというものを使っています。

```
cat.html
<!DOCTYPE html>
<html>
  <head>
    <title>Cat API Example</title>
  </head>
  <body>
    <img id="cat-image" src="" alt="Random cat image">
    <script>
      const request = new XMLHttpRequest();
      request.open('GET', 'https://api.thecatapi.com/v1/images/search');
      request.onload = function () {
        const data = JSON.parse(request.responseText);
        const catImage = document.getElementById('cat-image');
        catImage.src = data[0].url;
      };
      request.send();
    </script>
  </body>
</html>
```

• APIを呼ぶときの  
パラメーター

• Getを使って  
APIにリクエスト

• 応答をjson形式に変換

## 第3章 まとめ

本章では、無料でアクセス可能なWeb APIの一つである、The Cat APIに実際にWebブラウザからアクセスしデータを取得した。

JavaScriptフレームワークによるWebプログラミング  
第8回 Web API連携

## 第3章

# Web APIにアクセスしてみる

終わり

JavaScriptフレームワークによるWebプログラミング  
第8回 Web API連携

## 第4章

# Node.jsからWeb APIの利用

## 第4章 学習目標

- Web APIにNode.js Expressからアクセスするプログラムの作成ができる

# node.jsからThe Cat APIの実行

- 以下のコマンドでcatディレクトリを作成  
その下にcdで移動
  - % mkdir cat
  - % cd cat
- そこで以下のコードを、cat.jsという名前で作成、保存する。

cat.js

```
const request = require('request');

request('https://api.thecatapi.com/v1/images/search', function (error, response, body) {
  if (!error && response.statusCode === 200) {
    const data = JSON.parse(body);
    console.log(body);
    const catImageUrl = data[0].url;
    console.log(catImageUrl);
  }
});
```

# node.jsからThe Cat APIの実行

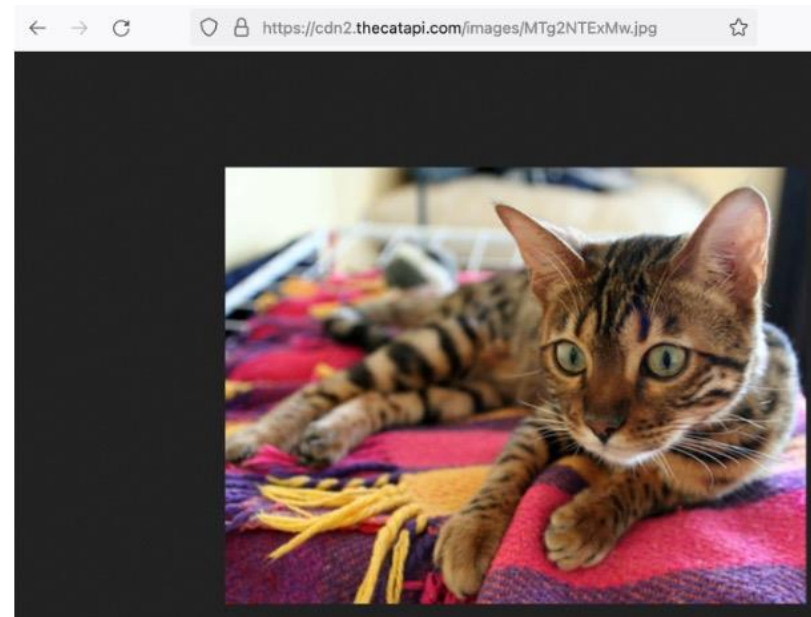
- terminalで、以下を実行

```
% npm install request
```

```
% node cat.js
```

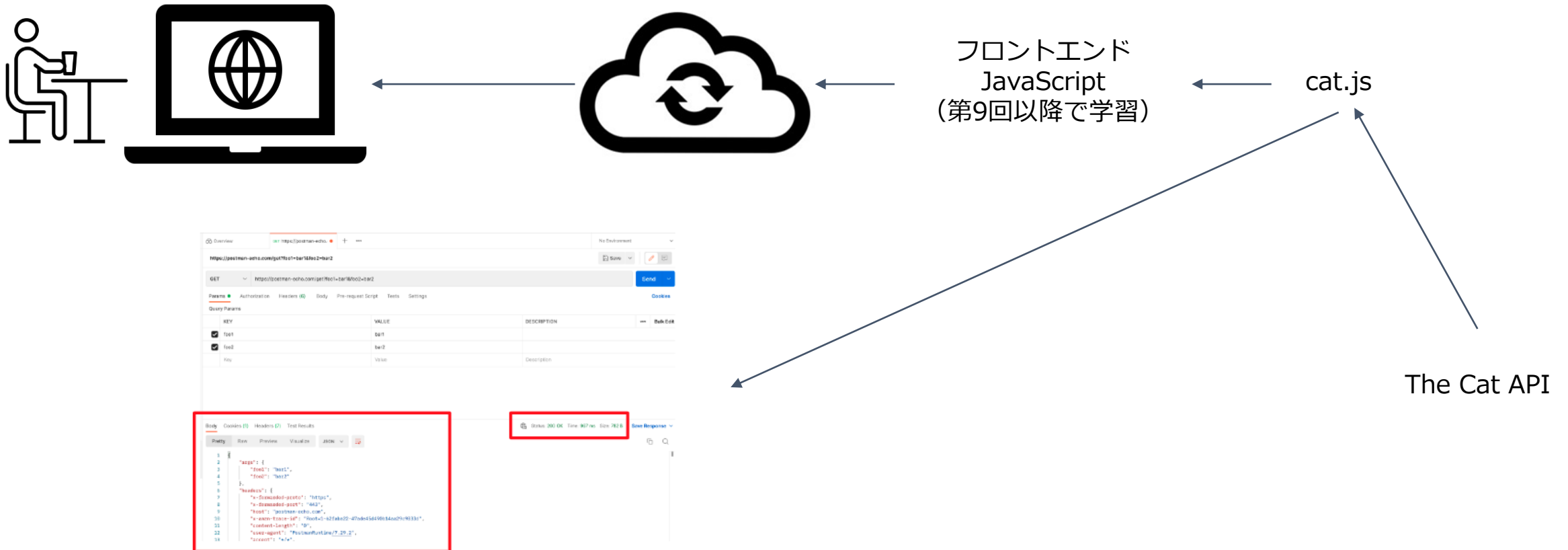
- 実行結果のURLをコピーし、ブラウザで表示
  - 猫の画像が表示される

```
[  
  {  
    id: 'MjAyODE3NQ',  
    url: 'https://cdn2.thecatapi.com/images/MjAyODE3NQ.jpg',  
    width: 611,  
    height: 593  
  }  
]  
(base) cat %
```



# The Cat APIとの連携

今回はThe Cat APIのデータをRest APIで応答するサービスを作成





# プログラムの場所について

プログラムは前回、第6回まで作成した場所（myapp）に作成

（ご自分のホームディレクトリなど） - myapp - （第4章、今回のプログラム）

|                   ↑  
                  ● またこのフォルダーに戻って実習

-- express\_mongodb - （第3章のプログラム）

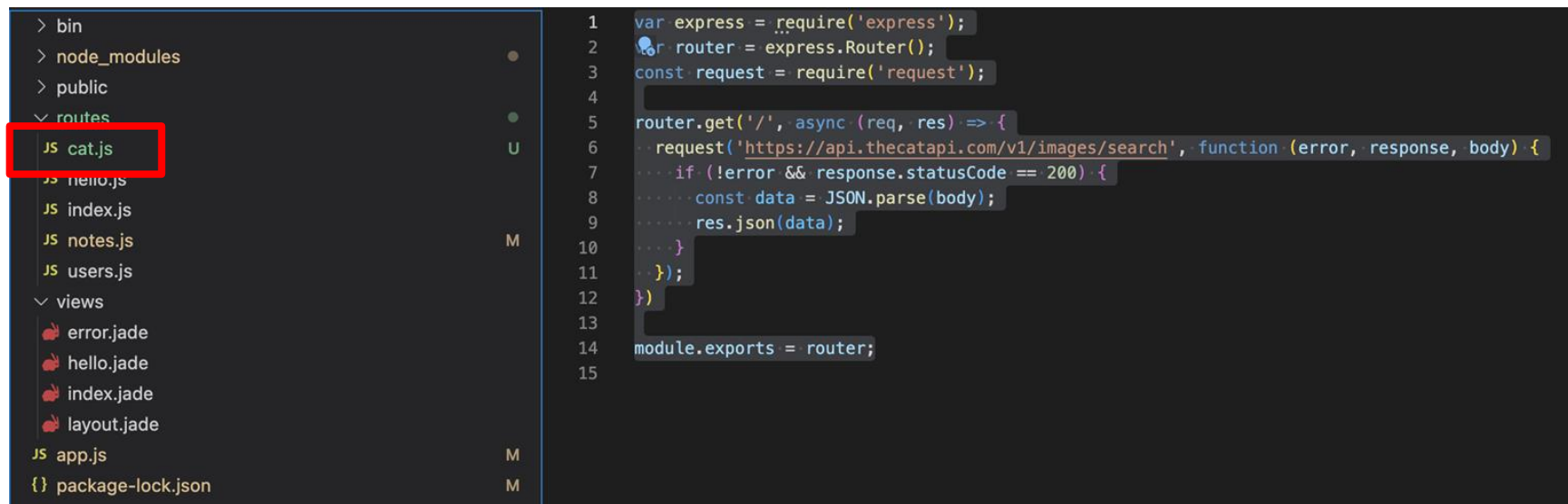
# cat.js作成

## routes配下にcat.jsを作成

```
var express = require('express');
var router = express.Router();
const request = require('request');

router.get('/', async (req, res) => {
  request('https://api.thecatapi.com/v1/images/search', function (error, response,
  body) {
    if (!error && response.statusCode === 200) {
      const data = JSON.parse(body);
      res.json(data);
    }
  });
});
```

```
module.exports = router;
```



# app.jsの修正

この2行を追加

```
var catRouter =  
require('./routes/cat');
```

```
app.use('/cat',  
catRouter);
```

```
7-3-EXPRESS
├── bin
│   └── www
├── node_modules
├── public
├── routes
│   ├── cat.js
│   ├── hello.js
│   ├── index.js
│   ├── notes.js
│   └── users.js
├── views
└── app.js
package-lock.json
package.json
```

```
JS app.js > ...
1  var createError = require('http-errors');
2  var express = require('express');
3  var path = require('path');
4  var cookieParser = require('cookie-parser');
5  var logger = require('morgan');
6
7  var indexRouter = require('./routes/index');
8  var usersRouter = require('./routes/users');
9  var helloRouter = require('./routes/hello');
10 var notesRouter = require('./routes/notes');
11 var catRouter = require('./routes/cat');
12
13 var app = express();
14
15 // view engine setup
16 app.set('views', path.join(__dirname, 'views'));
17 app.set('view engine', 'jade');
18
19 app.use(logger('dev'));
20 app.use(express.json());
21 app.use(express.urlencoded({ extended: false }));
22 app.use(cookieParser());
23 app.use(express.static(path.join(__dirname, 'public')));
24
25 app.use('/', indexRouter);
26 app.use('/users', usersRouter);
27 app.use('/hello', helloRouter);
28 app.use('/notes', notesRouter);
29 app.use('/cat', catRouter);
30
```

# ブラウザでの動作確認

- コマンドで以下を入力  
`% npm install request`
- npm startでアプリを起動し、ブラウザに、以下を入力。  
<http://localhost:30000/cat>
- 以下のように表示されたらOK。  
(\*ポート番号は自分のものを指定)



```
[{"id":"b2n","url":"https://cdn2.thecatapi.com/images/b2n.jpg","width":4608,"height":3456}]
```

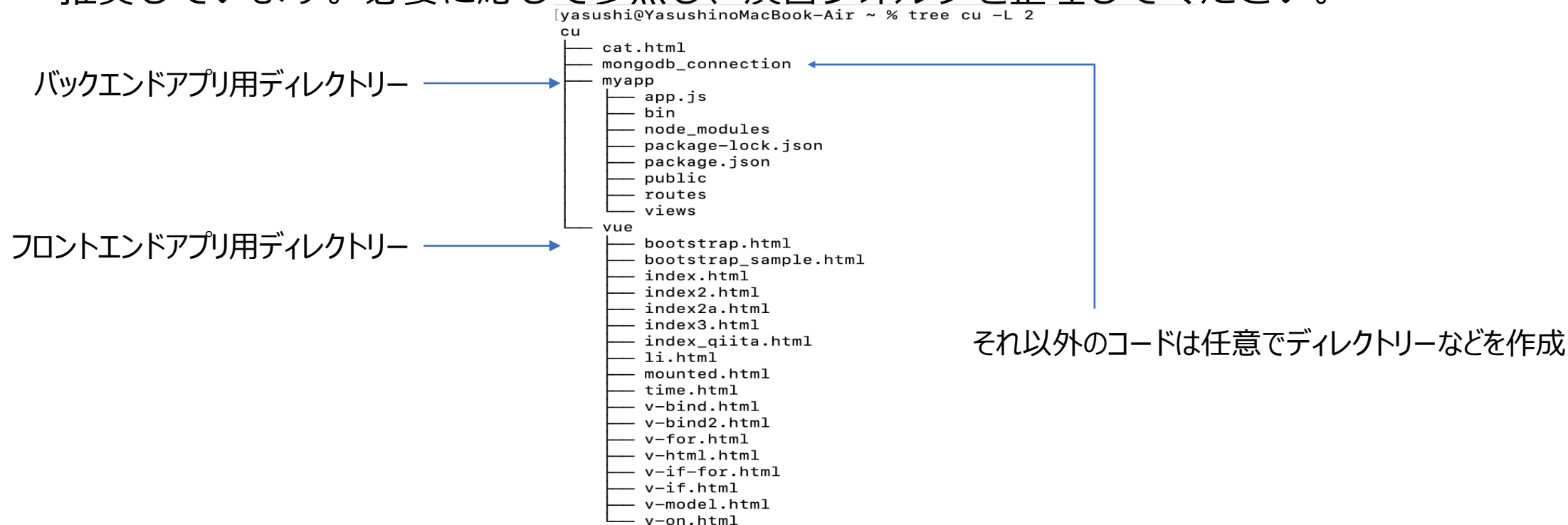
# 実機演習：The Cat APIとの連携

動画を全画面で視聴してください

# 実習時間

- ・ 10分程度を目安に動画を止めて前ページまでの実習をしてください。
- ・ 作業が終わったらビデオを再開して学習を進めてください。

※第1回4章で説明した通り、実習のファイル作成時には以下のディレクトリ構造を推奨しています。必要に応じて参照し、演習フォルダを整理してください。



# サーバでの動作確認

- GitHubへのコードのプッシュ
- サーバへのログイン
- Git pullの前にGit stashを行う
- Git Pullでサーバのコードを最新に
- サーバでアプリ起動
- Postmanで動作確認

アプリの確認のためのURLは  
(自分のサーバのIPアドレス:ポート番号/cat)  
例: 153.120.121.157:30000/cat

# 実機演習：サーバでの動作確認

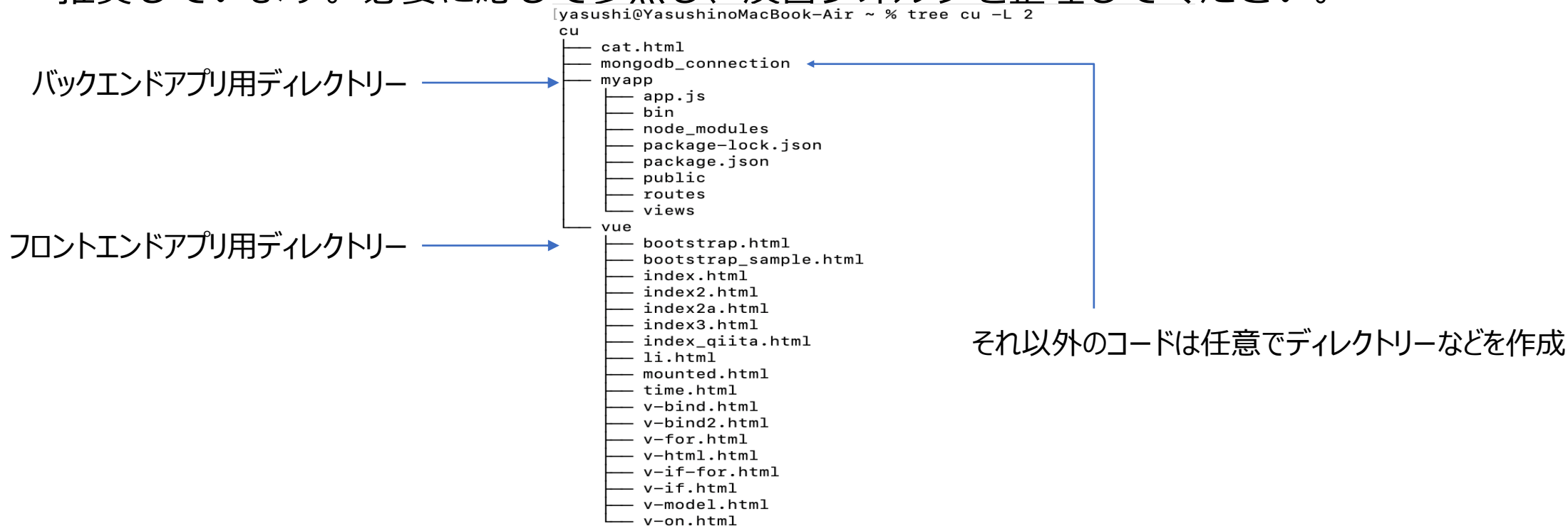
動画を全画面で視聴してください



# 実習時間

- ・ 10分程度を目安に動画を止めて前ページまでの実習をしてください。
- ・ 作業が終わったらビデオを再開して学習を進めてください。

※第1回4章で説明した通り、実習のファイル作成時には以下のディレクトリ構造を推奨しています。必要に応じて参照し、演習フォルダを整理してください。



## 第4章 まとめ

この章では、node.jsからThe Cat APIを利用し、猫の画像のリンクを取得し、ブラウザで猫の画像を表示した。またNode.js Expressからアクセスするプログラムを作成した。

## 第8回 まとめ

- Node.jsとMongoDBの連携プログラムを作成、動作確認を行った
- 無料で使用可能なWeb APIの中からいくつかの概要を学習した
- 無料でアクセス可能なWeb APIの一つである、The Cat APIに実際にWebブラウザからアクセスしデータを取得した
- そのAPIにNode.js Expressからアクセスするプログラムを作成した

JavaScriptフレームワークによるWebプログラミング  
第8回 Web API連携

# 第4章

## Node.jsからWeb APIの利用

### 終わり