

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開発

第4回 Express実習1

小園井康志

第4回 学習目標

- JavaScriptバックエンドフレームワークNodejs expressについて理解し説明できる
 - サーバを使ってExpressの実行環境を設定しクラウドのデプロイできる
- * サーバはさくらインターネットのサーバを利用する

第4回 授業構成

- 第1章 Express概要および設定
- 第2章 Express実行環境設定 (さくらインターネットのサーバ)

JavaScriptフレームワークによるWeb開発

第4回 Express実習1

第1章

Express概要および設定

第1章 学習目標

- JavaScriptバックエンドフレームワークExpressについて理解し説明できる
- Express Generatorを使ってプロジェクトを作成することができる

Nodejs Express概要

- Express

Nodejsに機能を追加してアプリケーション開発をより簡単に行えるようにしたアプリケーションフレームワーク

Nodejs本来の機能を活かした比較的小さなフレームワーク

GET/POST通信などを簡単な記述で実現できる。

最近はREST APIの開発に使用されることが多い。（バックエンド）

Nodejs Express概要

- Express Generator

Expressには用意されている、Webアプリケーションの雛形、プロジェクトを作成してくれるツール。

- アプリケーションの基本部分を自動生成する。
- 必要なパッケージが入っている。

Express Generatorのインストール

- ・コマンド画面で以下のように入力

```
% npm install -g express-generator
```

- ・プロジェクトの作成

```
% express myapp
```

- ・依存関係をインストールして実行

```
% cd myapp
```

```
% npm install
```

```
% npm start
```

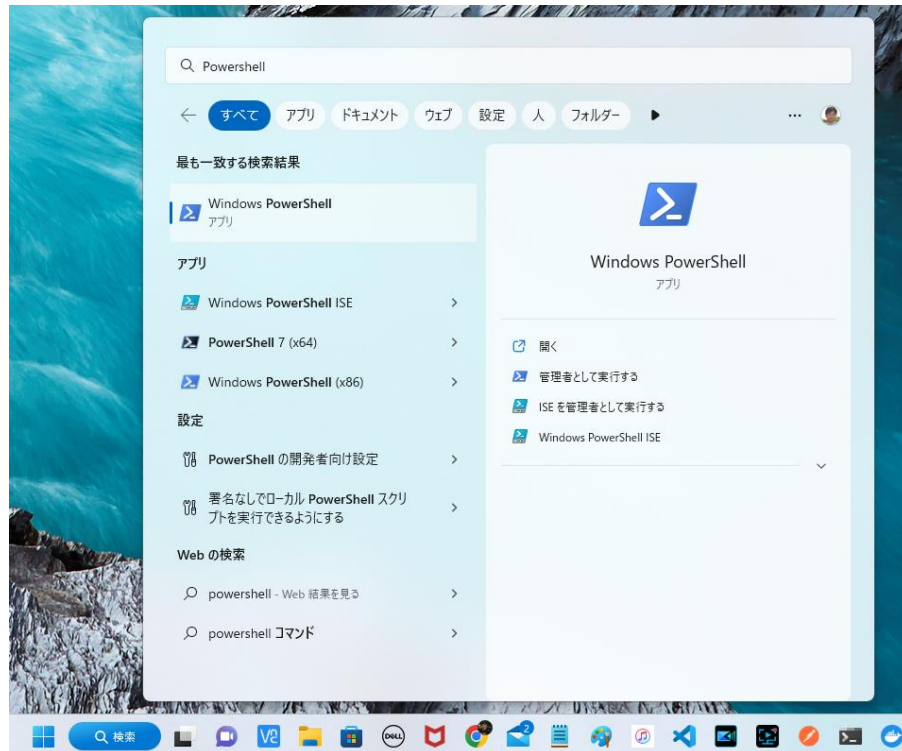
- ・ブラウザを立ち上げて<http://localhost:3000> をアクセス

* Windowsの場合権限の関係上左記のインストールで失敗する場合があります。
その場合インストール作業をAdministrator権限で行なってください。（次ページ参照）

Express Generatorのインストール（Windows編）

- Administrator権限でのPowerShellの起動

② Powershellと入力



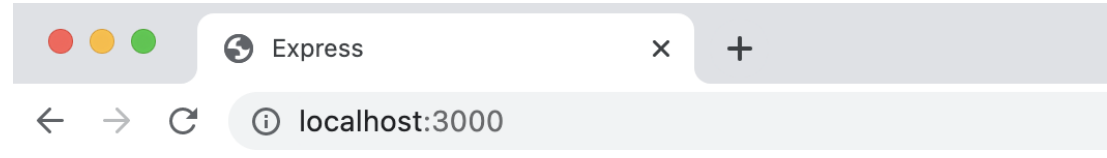
① 検索をクリック



Windows PowerShell上で右クリック □ 管理者で実行を選択

Express Generatorのインストール

- ・動作画面の確認



Express

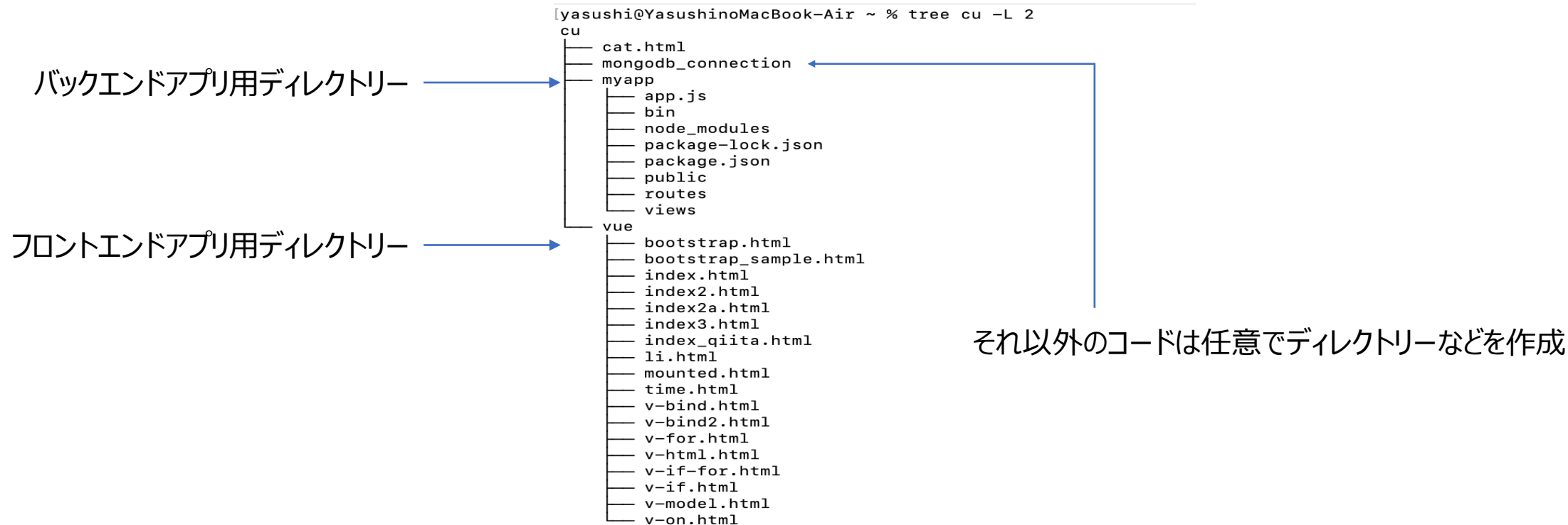
Welcome to Express

- ・確認できたら"Ctrl + C"でプログラムを終了

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。



Express Generator補足説明

- ・ npm installで行っていること
- ・ プロジェクトに必要なパッケージをインストール
- ・ 必要なパッケージは package.json に記載されているもの
- ・ インストール先は node_modules フォルダ

```
[sample-app % ls
```

```
app.js
```

```
bin
```

```
node_modules
```

```
package-lock.json
```

```
package.json
```

```
public
```

```
routes
```

```
views
```

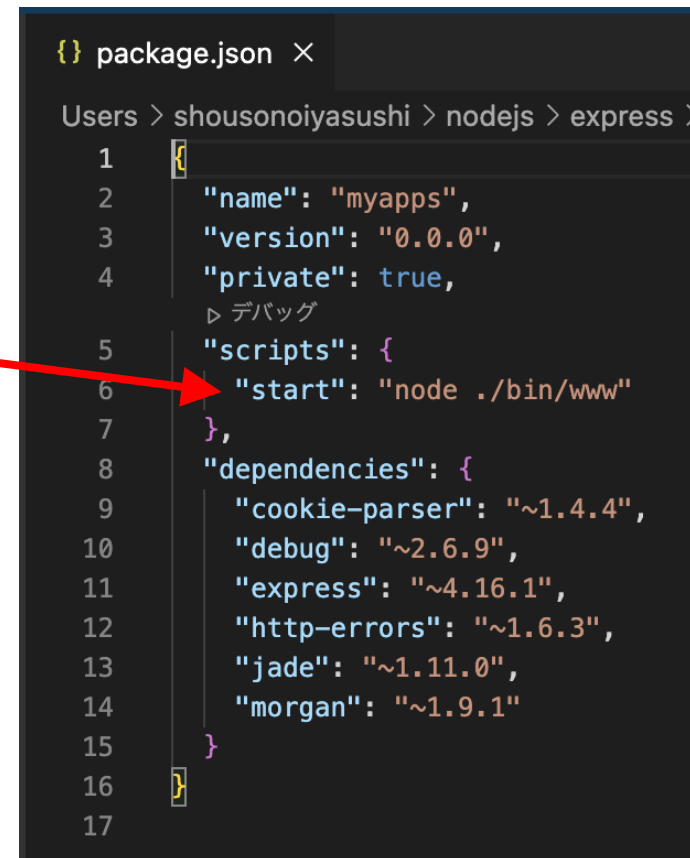
Express Generator補足説明

- Npm startで行っていること
- Bin フォルダの wwwを実行

`node bin/www`

と同等

- 通常はpackage.jsonの中にstartコマンドで実行することを記述

A screenshot of a code editor window titled "package.json". The editor shows the following JSON content:

```
{
  "name": "myapps",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "cookie-parser": "~1.4.4",
    "debug": "~2.6.9",
    "express": "~4.16.1",
    "http-errors": "~1.6.3",
    "jade": "~1.11.0",
    "morgan": "~1.9.1"
  }
}
```

A red arrow originates from the text "実行することを記述" in the list above and points to the "start" property in the "scripts" object of the JSON.

Express Generator補足説明

・アプリケーションのフォルダ構成

```
.
├── app.js
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
│       └── style.css
├── routes
│   ├── index.js
│   └── users.js
└── views
    ├── error.jade
    ├── index.jade
    └── layout.jade
```

Express Generator補足説明

- “bin” フォルダ
 - ・ アプリケーションを実行するためのコマンドファイルが保管されている
 - ・ 今回はwwwという実行ファイルがありこれが今回の実行ファイル
- “public” フォルダ
 - ・ 公開用ディレクトリ、ウェブアプリケーションで使うイメージ画像、スクリプトファイル、スタイルシートが置かれている
- “route” フォルダ
 - ・ ルーティング（どのページにアクセスするか）を追加するためのスクリプトファイルが置かれている

Express Generator補足説明

- “view” フォルダ
 - ページを表示するときのテンプレートファイルが置かれている。
- package.jsonについて
 - アプリに必要なライブラリがリストされている。
 - それらは“npm install”実行時に“node_module”フォルダ配下にダウンロードされる

```
() package.json ×
Users > shousonoiyasushi > nodejs > express > temp > myapps > () package.json > ...
1  {
2    "name": "myapps",
3    "version": "0.0.0",
4    "private": true,
5    "scripts": {
6      "start": "node ./bin/www"
7    },
8    "dependencies": {
9      "cookie-parser": "~1.4.4",
10     "debug": "~2.6.9",
11     "express": "~4.16.1",
12     "http-errors": "~1.6.3",
13     "jade": "~1.11.0",
14     "morgan": "~1.9.1"
15   }
16 }
17
```

```
[myapps % ls node_modules
accepts          cookie           http-errors      morgan           setprototypeof
acorn            cookie-parser    iconv-lite       ms               source-map
acorn-globals    cookie-signature inherits          statuses
align-text       css              ipaddr.js        negotiator       transformers
amdefine         css-parse       is-buffer        on-finished     type-is
array-flatten    css-stringify   is-promise       on-headers      uglify-js
asap             debug           jade             parseurl        uglify-to-browserify
basic-auth       decamelize      jstransformer    path-to-regexp  unpipe
body-parser      depd            kind-of          promise         utils-merge
bytes            destroy         lazy-cache       proxy-addr      vary
camelcase        ee-first        longest          qs              void-elements
center-align     encodeurl       media-typer      range-parser    window-size
character-parser escape-html     merge-descriptors raw-body        with
clean-css        etag            methods          repeat-string   wordwrap
cliui            express         mime             right-align     wrap
commander        finalhandler    mime-db          safe-buffer     yargs
constantinople  forwarded      mime-types       safer-buffer
content-disposition fresh           minimist        send
content-type     graceful-readlink mkdirp          serve-static
```



ソースコードをGitHubにアップロード

- GitHubにリポジトリを作成する
 - GitHubにブラウザでログイン
 - Newをクリックレポジトリを追加

Recent Repositories



Find a repository...

 osonoi2019/test

 osonoi2019/nodejs_training

Recent activity

When you take actions across GitHub, we'll provide links to that activity here.

ソースコードをGitHubにアップロード

- ・ 任意のレポジトリ名を入力
- ・ （ここではexpress）
- ・ “Create Repository” をクリック

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * osonoi2019 / Repository name * express ✓

Great repository name: [Learn more about refactored-barnacle?](#)

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: Apache License 2.0

① You are creating a public repository in your personal account.

Create repository

ソースコードをGitHubにアップロード

- Expressのディレクトリに移動し、以下のコマンドを入力

```
% git init
```

```
% git add .
```

```
% git commit -m "first commit"
```

```
% git branch -M main
```

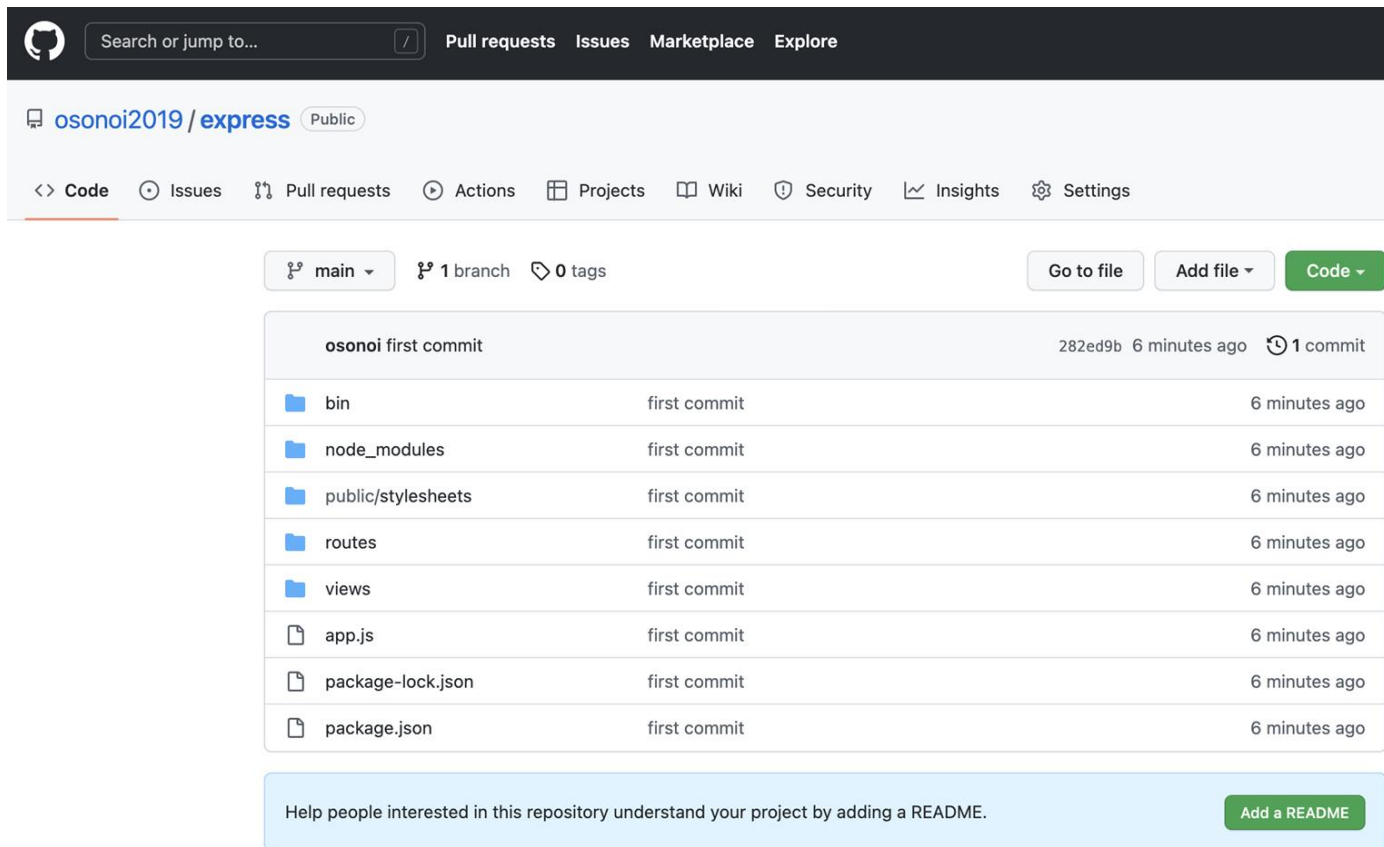
```
% git remote add origin https://github.com/\(自身のアカウント\)/express.git
```

```
% git push origin main
```

以上でPCのソースコードがGitHubに反映される

ソースコードをGitHubにアップロード

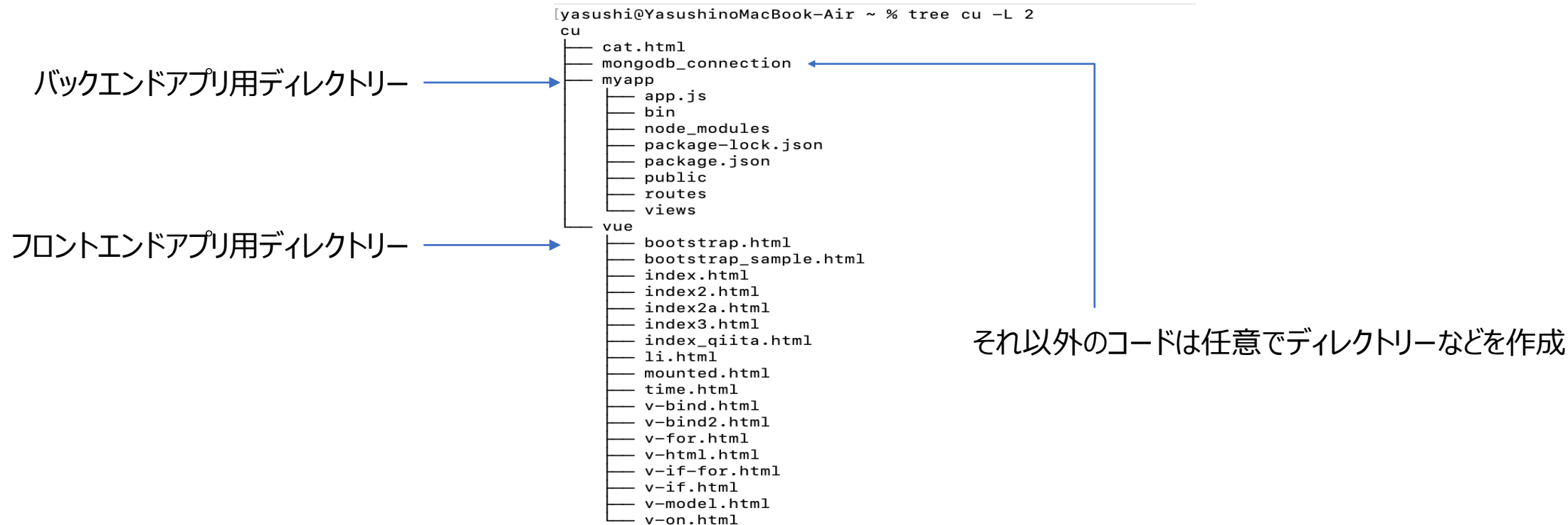
- GitHubのサイトで確認



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1章 まとめ

- JavaScriptバックエンドフレームワークNodejs expressについて学習をした
 - express : Nodejsに機能を追加して、アプリケーション開発をより簡単に行えるようにしたアプリケーションフレームワーク
 - Express Generator : expressに用意されている、Webアプリケーションの雛形、プロジェクトを作成してくれるツール
- Express Generatorを使ったプロジェクトの作成方法を学習した
 - Express Generatorのインストール
 - expressによるプロジェクトファイルの作成
 - Githubへのソースコードのアップロード

JavaScriptフレームワークによるWeb開発

第4回 Express実習1

第1章

Express概要および設定

終わり

JavaScriptフレームワークによるWeb開発

第4回 Express実習1

第2章

Express実行環境設定（サーバ編）

第2章 学習目標

- インターネット上のサーバを使ってExpressの実行環境を設定しクラウドにWebサイトをデプロイできる

今回使用するサーバについて

今回の演習ではインターネット上のサーバを使って作成したプログラムを実行していただきます。

使用するサーバはさくらインターネットのVPSサーバというサービスを使う。

<https://vps.sakura.ad.jp/>

サーバに接続するため、事前に配布した以下の情報を確認してください。

(サーバのIP アドレス、ポート番号、ユーザー名、パスワード)

サーバで動作確認する前にソースコードの編集（ポート番号の変更）

- ・ 第1章で作成したプログラムは3000番ポートで待ち受ける形になっている
- ・ 今回共有サーバを使っているので各自の番号のポート番号で待ち受けるように変更する必要がある
- ・ 変更するファイル
- ・ binフォルダーの中のwww
- ・ ローカルPCのコードを変更、その後Githubにプッシュ

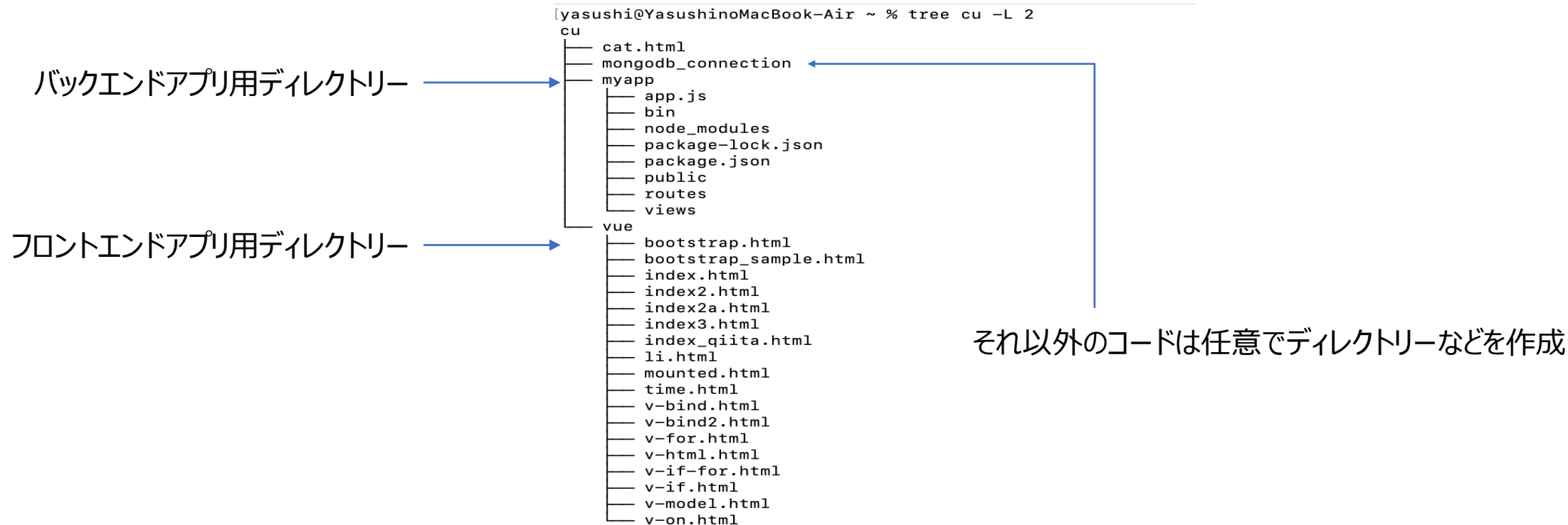
```
/**  
 * Get port from environment and store in Express.  
 */  
var port = normalizePort(process.env.PORT || '3000');  
app.set('port', port);
```

この3000をご自分の
割り当てられた番号に変更

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。



サーバへのデプロイ手順

- ・サーバへのsshを使った接続
- ・GitHubソースコードをサーバにコピー（Clone）
- ・サーバ上の動作確認

サーバへのsshを使った接続

- ・ コマンドラインからsshツールを使ってサーバにログインする

```
% ssh (ユーザー名)@(IPアドレス)
```

この後パスワードの入力を求められるので入力

サーバへのsshを使った接続

- sshコマンドでサーバにログイン
- ssh (ユーザー名)@(サーバのIPアドレス)
- (パスワードを入力)

• 実行例

ご自分のユーザー名 サーバのIPアドレス

```
[[osonoi@ik1-228-81903 ~]$ ssh osonoi@153.120.121.157
The authenticity of host '153.120.121.157 (153.120.121.157)' can't be established.
ECDSA key fingerprint is SHA256:3XS1qErwMUOiV0H7x3MXDnfxg1jmcSmBfdtYctfseWs.
[Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '153.120.121.157' (ECDSA) to the list of known hosts.
[osonoi@153.120.121.157's password:
```

上記のメッセージが
出てきたら"yes"と入力

ここでパスワードを入力

```
SAKURA internet [Virtual Private Server SERVICE]
```

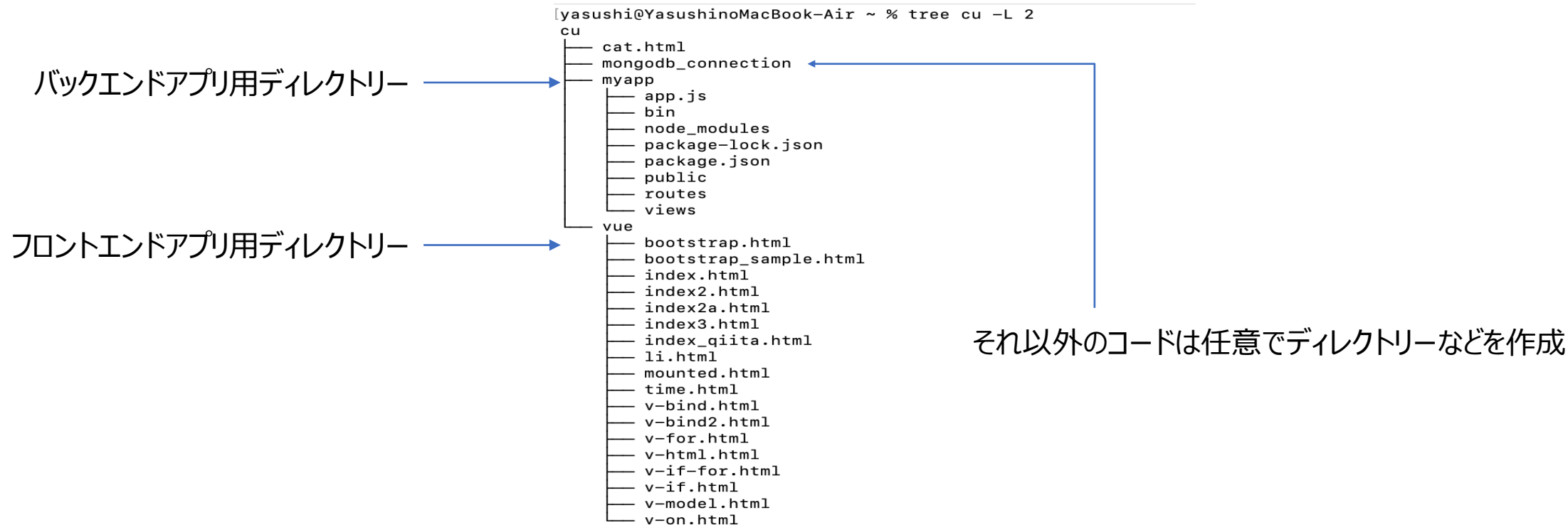
```
Last login: Mon Dec 26 16:04:59 2022 from 129.41.57.19
```

```
[osonoi@ik1-228-81903 ~]$ █
```

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。



GitHubソースコードをサーバにコピー（Clone）

- ・次のコマンドでGitHubのコードをサーバにCloneする

```
% Git clone (Githubのリポジトリのアドレス)
```

- ・実行例

```
[[osonoi@ik1-228-81903 ~]$ git clone https://github.com/osonoi-hot/express.git
Cloning into 'express'...
remote: Enumerating objects: 1076, done.
remote: Counting objects: 100% (1076/1076), done.
remote: Compressing objects: 100% (836/836), done.
remote: Total 1076 (delta 180), reused 1076 (delta 180), pack-reused 0
Receiving objects: 100% (1076/1076), 1.48 MiB | 2.00 MiB/s, done.
Resolving deltas: 100% (180/180), done.
[osonoi@ik1-228-81903 ~]$
```

プログラムを実行

- ・次のコマンドで実行

```
% cd express
```

```
% npm install
```

```
% npm start
```

- ・実行例

```
[[osonoi@ik1-228-81903 ~]]$ cd express
[[osonoi@ik1-228-81903 express]$ npm install

up to date, audited 100 packages in 2s

1 package is looking for funding
  run `npm fund` for details

8 vulnerabilities (1 low, 3 high, 4 critical)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
[[osonoi@ik1-228-81903 express]$ npm start

> myapp@0.0.0 start
> node ./bin/www

■
```

サーバ上の実行確認

ブラウザーで自分のIPアドレス：ポート番号でアクセス
第1章と同じ画面が出ることを確認



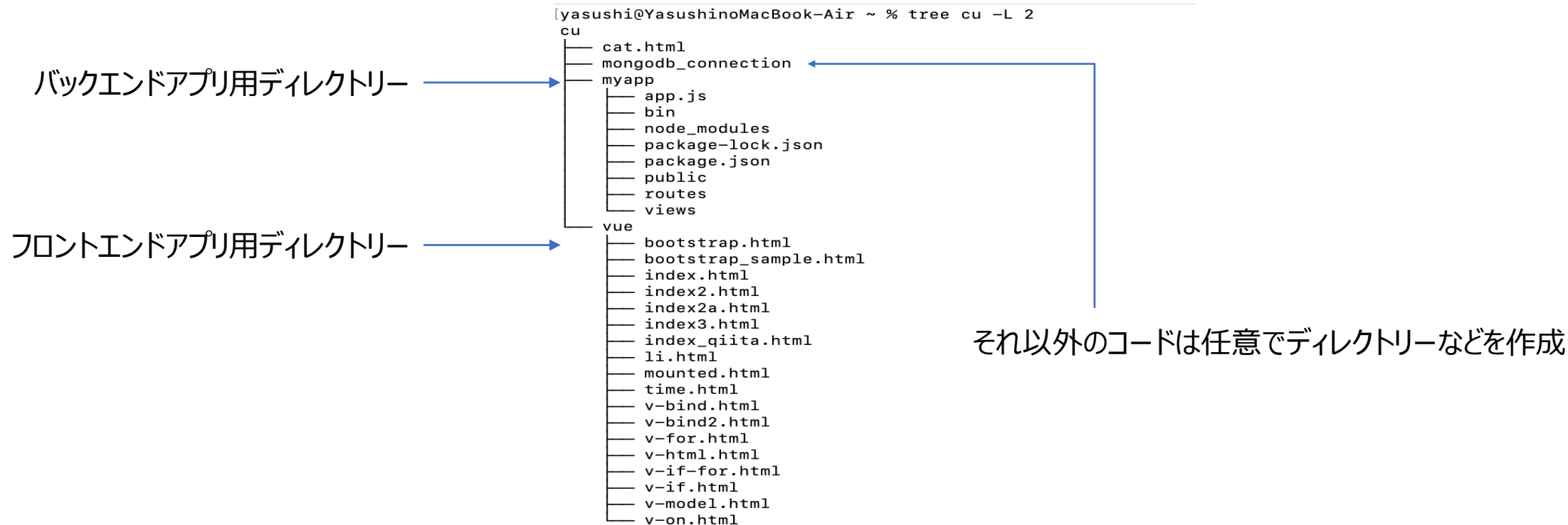
Express

Welcome to Express

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2章 まとめ

- GitHubを利用し、インターネット上のサーバにWebサイトをデプロイする方法を学習した。

- プログラムのポート番号を、使用するサーバのポート番号に変更
- sshコマンドを使用して、コマンドラインからサーバにアクセス
- gitCloneコマンドで、githubからサーバ上にプログラムをコピーし動作確認

第4回 まとめ

- JavaScriptバックエンドフレームワーク
Nodejs expressについて学習をした
- Express Generatorを使ったプロジェクトの
作成方法を学習した
- GitHubを利用し、インターネット上のサーバに
Webサイトをデプロイする方法を学習した。

JavaScriptフレームワークによるWeb開発

第4回 Express実習1

第2章

Express実行環境設定（サーバ編）

終わり