7. Webサーバのリクエスト

イベント

createServer()

`createServer()` でサーバを作成すると、コールバック関数が実行されました。コールバック関数の引数 `request` 、 `response` を使ってプログラミングします。

```
const app = http.createServer((request, response) => {

});
```

request.on()

`request.on()` でイベントを登録します。イベント発生後のコールバック関数を設定します。

```
request.on(イベント名, コールバック)
```

date イベント

`data` イベントはデータ受信時に発生するイベントです。コールバック関数の引数に受信したデータが入ります。

```
request.on('data', (value) => {
    //処理
});
```

endイベント

`end` イベントは、リクエスト完了後に処理するイベントです。

```
request.on('end', () => {
    //処理
});
```

プロジェクトの準備

HTMLファイルを読み込んでフォームを表示し、Webサーバにリクエスト＆レスポンスをしてみます。

モジュール追加

querystring

querystring はクエリ（文字列のURL）をオブジェクトに変換するモジュールです。

fs

fs はファイル操作に関するモジュールです。外部ファイルを読み込むために利用します。

dotenv

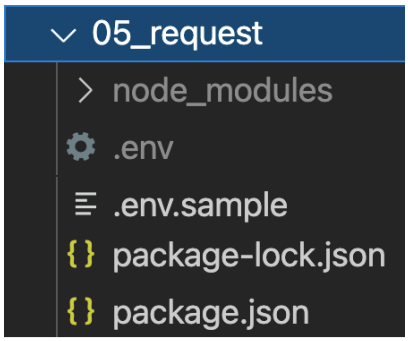
dotenv は .env ファイルの設定を読み込んで変数に設定します。

ターミナルで fs 、 querystring 、 dotenv モジュールをインストールします。

```
% npm i querystring fs dotenv
```

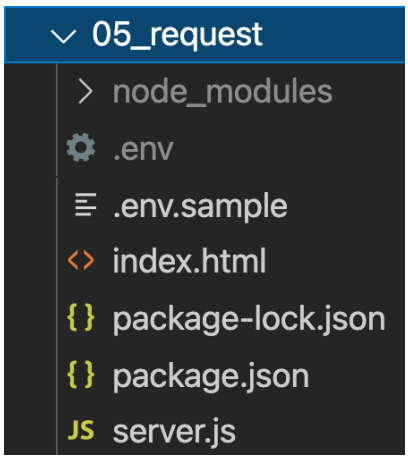
モジュール作成確認

モジュールは node_modules/ にインストールされ、パッケージ管理設定ファイル package-lock.json 、 package.json も自動で作成されます。



ファイル構成

`.env` 、 `index.html` 、 `server.js` ファイルを作成してプログラミングします。



.env の作成

`.env` ファイルを作成し、 `HOST` と `PORT` を設定します。

```
HOST=localhost
PORT=3000
```

HTMLフォーム作成

`index.html` を作成し、メールアドレスとパスワードを `POST` 送信するフォームを作成します。

index.html

```
<form action="" method="post">
  <p>Email：<input type="text" name="email"></p>
```

```
    <p>PW：<input type="password" name="password"></p>
    <button>Login</button>
  </form>
```

サーバの準備

`server.js` を作成してサーバを構築していきます。まず各モジュールを読み込みしてみましょう。

httpモジュール読み込み

`http` モジュールを読み込みます。

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

クエリデータの受信

`querystring` モジュールを読み込んで、クエリデータを受信します。

```
const querystring = require('querystring');
let obj = querystring.parse(query);
```

HTML読み込み

`fs` モジュールを使って `readFileSync()` で `index.html` を変数に読み込みます。

```
const fs = require('fs');
const html = fs.readFileSync('index.html');
```

.env の読み込み

`dotenv` モジュールの `config()` メソッドで `.env` ファイルを読み込みます。モジュールを読み込むと `process.env` が利用できるので、ホストとポートを読み込んで定数にします。

```
require('dotenv').config();
const host = process.env.HOST;
```

```
const port = process.env.PORT;
```

サーバ作成

ヘッダ出力

`http.createServer()` でサーバを作成し、 `200` ヘッダーを出力します。

```
const app = http.createServer((request, response) => {
    response.writeHead(200, {'Content-Type': 'text/html'});
}
```

dataイベント

`request.on()` で `data` イベントを登録し、コールバック関数で受信データをキャッシュしておきます。

```
const app = http.createServer((request, response) => {
    response.writeHead(200, {'Content-Type': 'text/html'});

    let post = '';
    request.on('data', (value) => {
        //受信データをキャッシュ
        post+= value;
    });

}
```

endイベント

`request.on()` の `end` イベントでレスポンス処理してHTMLを出力します。また、受信したデータをオブジェクトに変換してログに出力します。

```
...
    request.on('end', () => {
        if (post) {
            post = querystring.parse(post);
            console.log(post);
        }
        response.end(html);
```

```
    });
...
```

ログ

リクエストのメソッドとURLもログに出力しておきます。

```
...
    console.log(`Method: ${request.method}`);
    console.log(`URL: ${request.url}`);
...
```

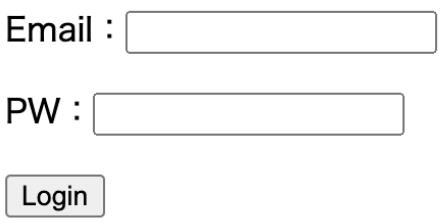
サーバ起動

ターミナルで `server.js` を実行してWebサーバを起動します。

```
% node server.js
Server listen: http://localhost:3000
```

データ送信

ブラウザで確認し、データを送信してみましょう。



ログの確認

`POST` 送信したデータを、ターミナルのログで確認してみましょう。

```
...
Method: POST
URL: /
```

```
[Object: null prototype] { email: 'tokyo@test.com', password: 'test' }
...
```

ソース

server.js

```
const http = require('http');
const querystring = require('querystring');
const fs = require('fs');
const html = fs.readFileSync('index.html');

require('dotenv').config();
const host = process.env.HOST;
const port = process.env.PORT;

const app = http.createServer((request, response) => {
    response.writeHead(200, {'Content-Type': 'text/html'});

    let post = '';
    request.on('data', (value) => {
        post+= value;
    });
    request.on('end', () => {
        if (post) {
            post = querystring.parse(post);
            console.log(post);
        }
        response.end(html);
    });

    console.log(`Method: ${request.method}`);
    console.log(`URL: ${request.url}`);
});

app.listen(port, host);

console.log(`Server listen: http://${host}:${port}`);
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <form action="" method="post">
    <p>Email：<input type="text" name="email"></p>
    <p>PW：<input type="password" name="password"></p>
    <button>Login</button>
  </form>
</body>
</html>
```