メディアアート・プログラミング2

東京藝術大学 芸術情報センター開設科目 後期金曜4限 第6週

2023.11.10 松浦知也（matsura.tomoya@noc.geidai.ac.jp teach@matsuuratomoya.com）

Node.jsでライブラリのインストール

他人のコードを使う：ライブラリ

- モジュール：コードをそれぞれの機能ごとに分割したもの
- ライブラリ：目的に合わせた再利用性の高いコードの集合体
- パッケージ：ライブラリやモジュールを再配布するためのメタデータ等が付随したもの
  - パッケージマネージャ：インターネット上の置き場：リポジトリからパッケージをダウンロード、アップロードしたりバージョン管理ができる仕組み

※Javascriptではライブラリという言葉を使うことはあまりなく、
言語ごとに使い分けも微妙に異なるのでざっくりした一般論くらいに思ってください

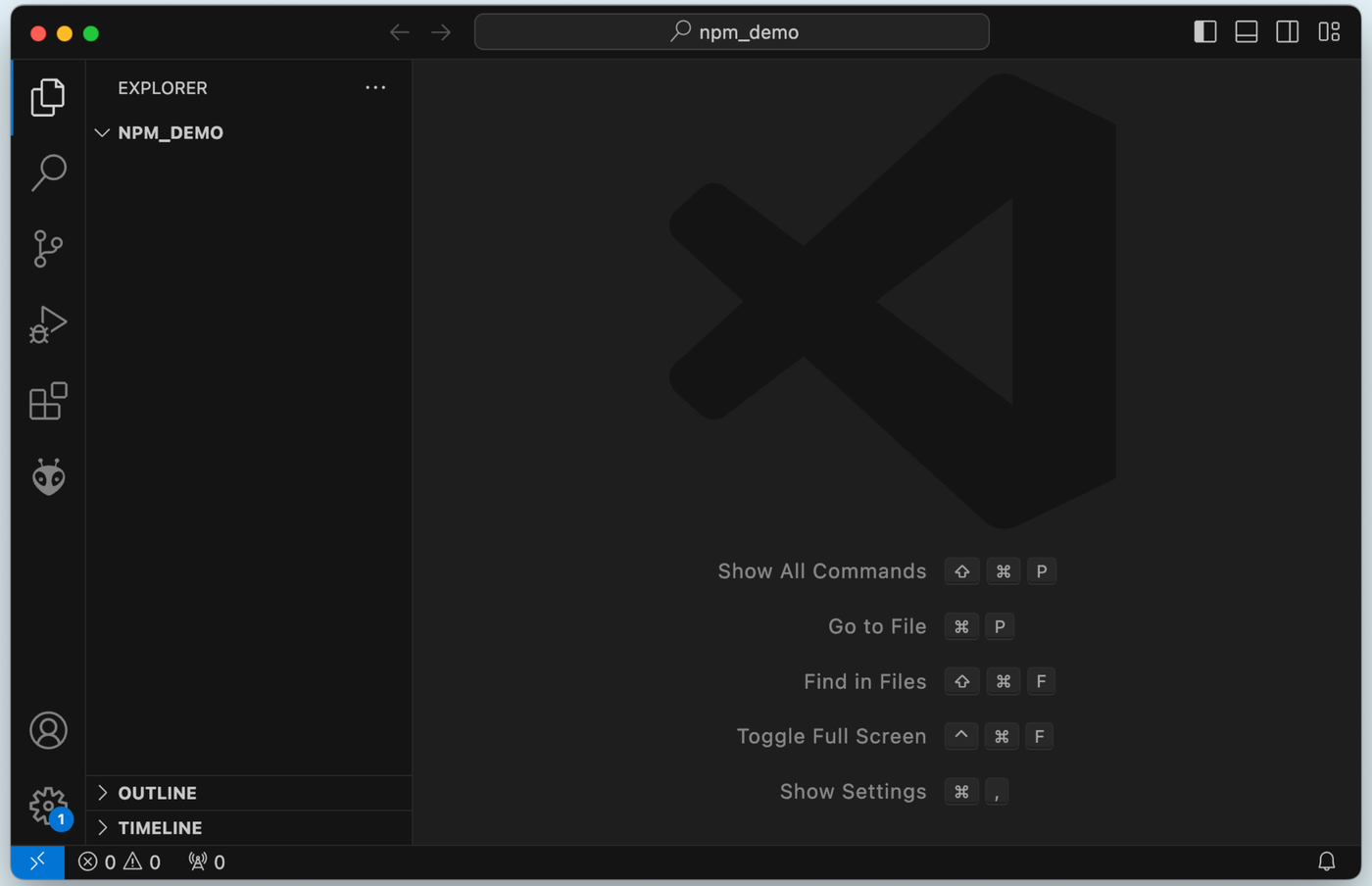
他人のコードを使う：ライブラリ

- モジュール：コードをそれぞれの機能ごとに分割したもの
- ライブラリ：目的に合わせた再利
- パッケージ：ライブラリやモジュ 付随したもの
  - パッケージマネージャ：インターネット上の置き場：リポジトリからパッケージをダウンロード、アップロードしたりバージョン管理ができる仕組み

例えばC++ではConanやvcpkgなどパッケージマネージャが個別のソフトウェアとして存在するが、比較的新しい言語にはセットでパッケージマネージャがついてくることも多い（Node.jsはnpm、Pythonはpipなど）

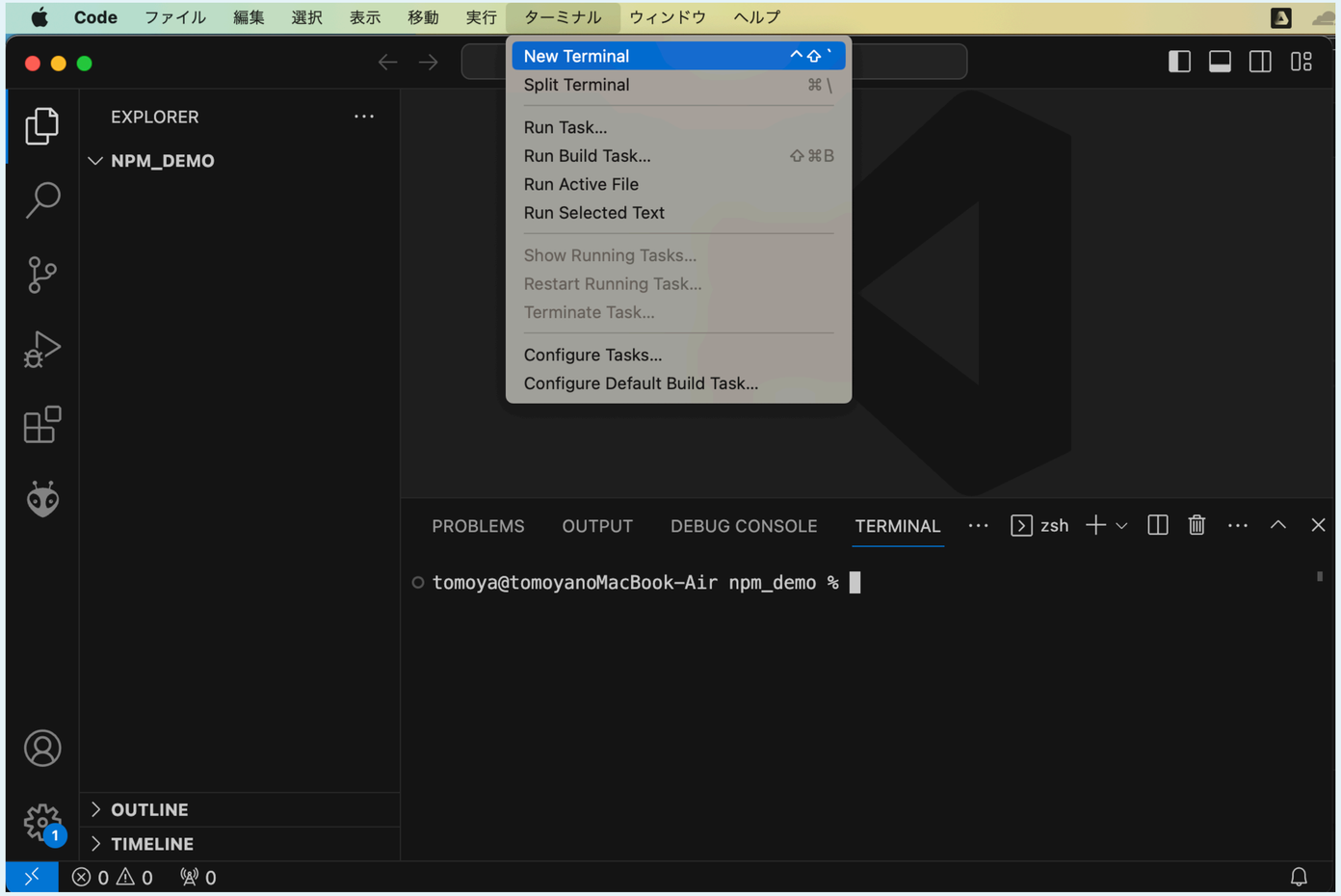
※Javascriptではライブラリという言葉を使うことはあまりなく、
言語ごとに使い分けも微妙に異なるのでざっくりした一般論くらいに思ってください

NPM(node package manager)を使ってみよう



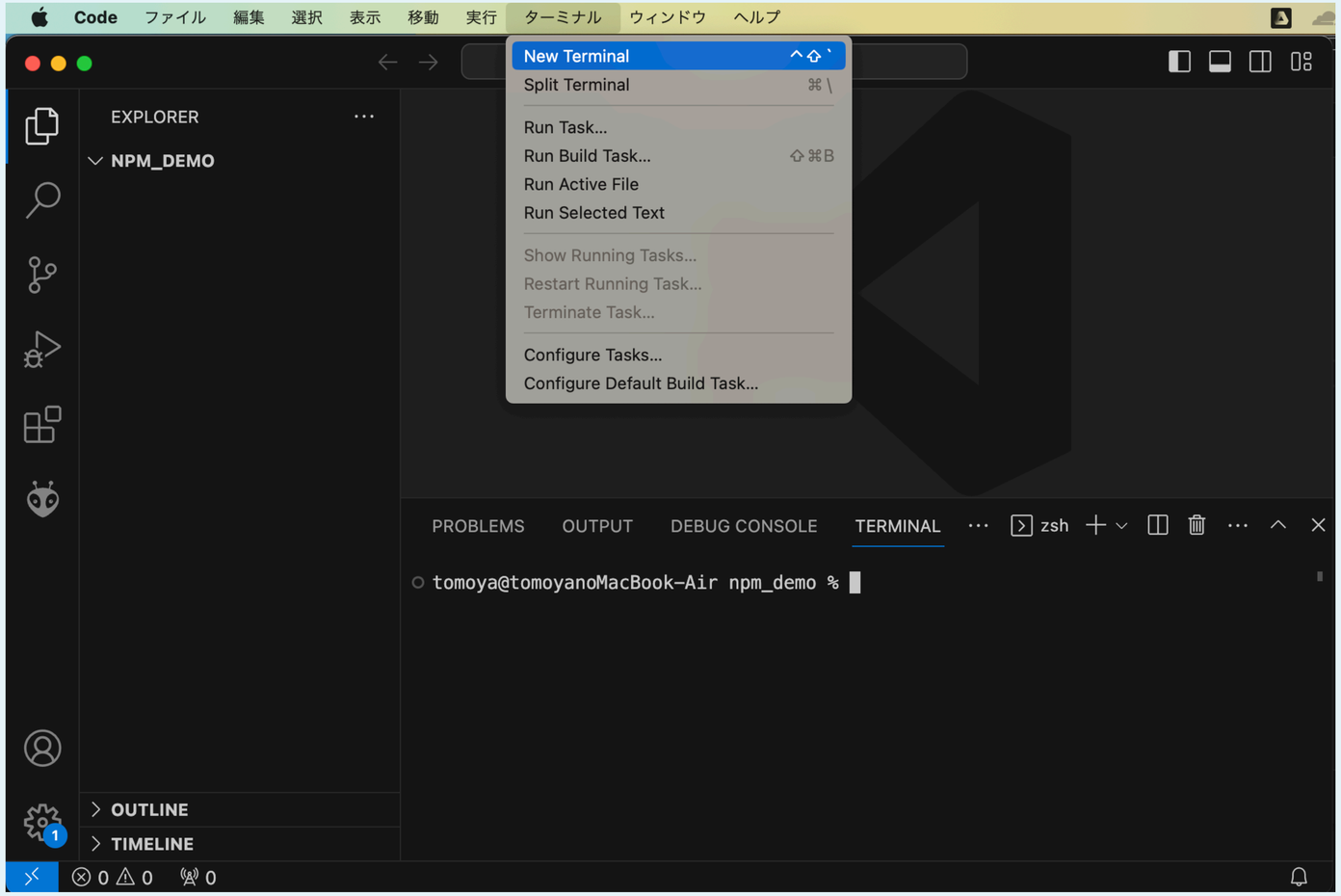
空のフォルダを作ってVSCodeで開く

NPM(node package manager)を使ってみよう



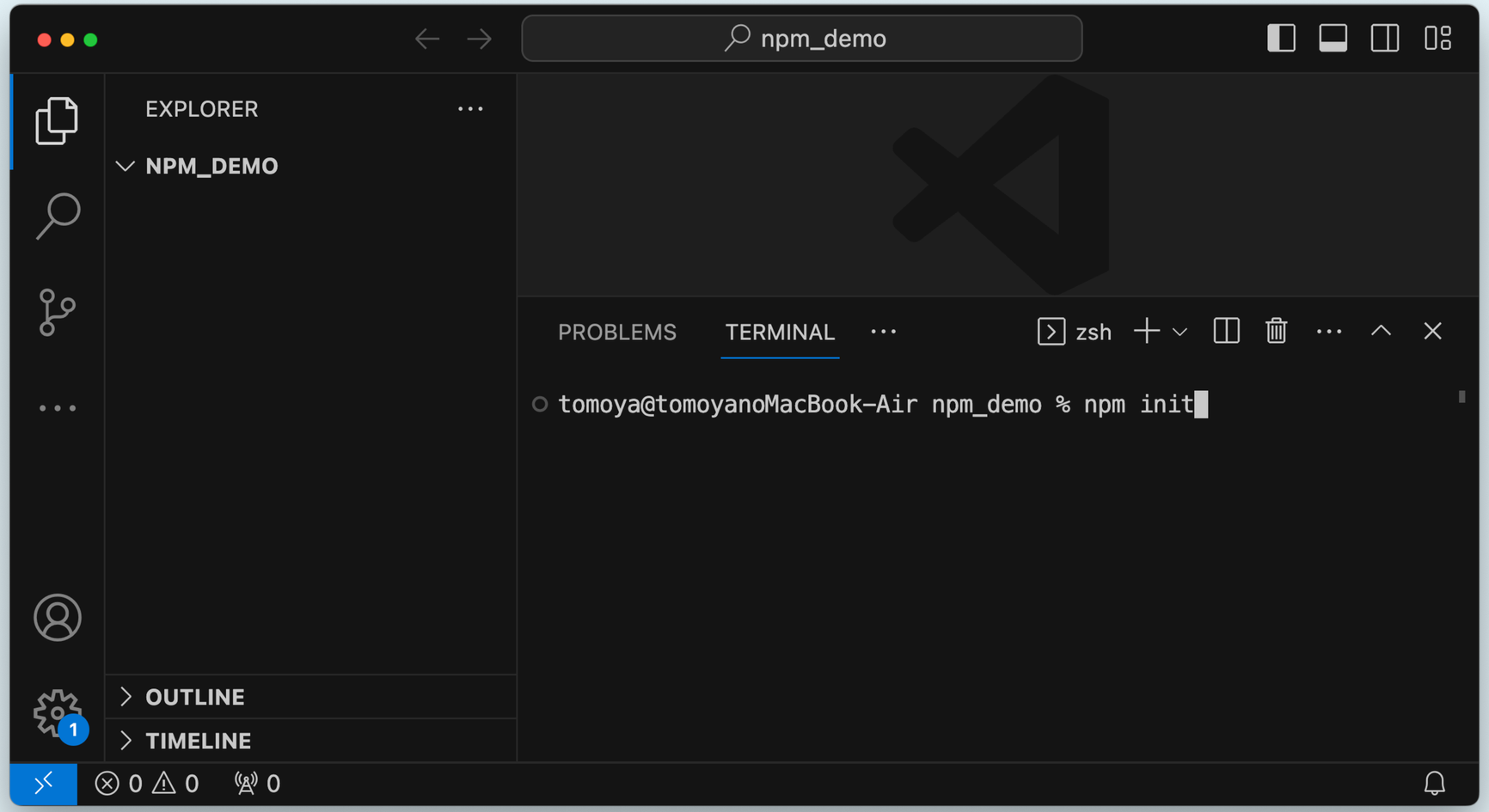
ターミナルを開く

NPM(node package manager)を使ってみよう



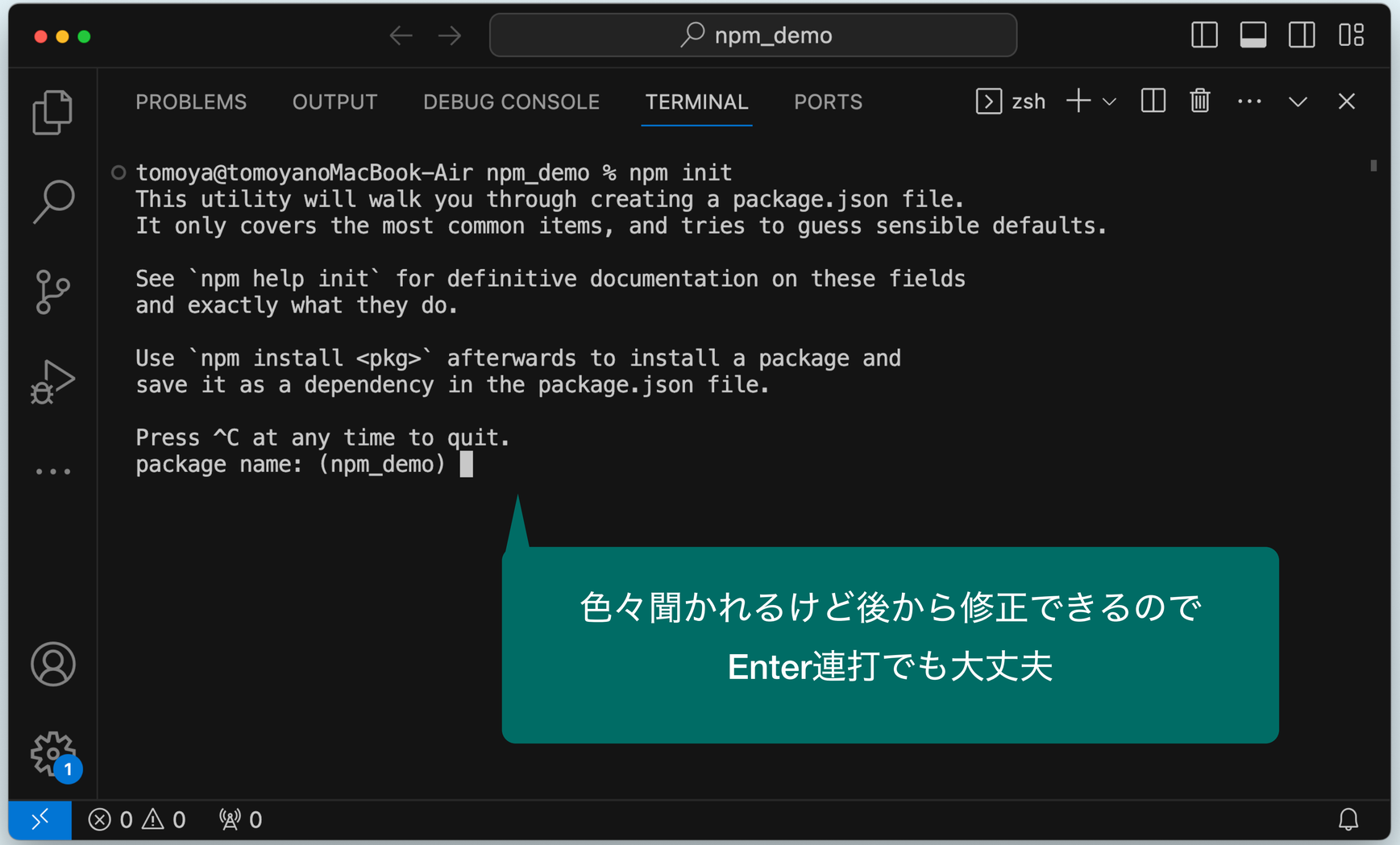
ターミナルを開く

NPM(node package manager)を使ってみよう

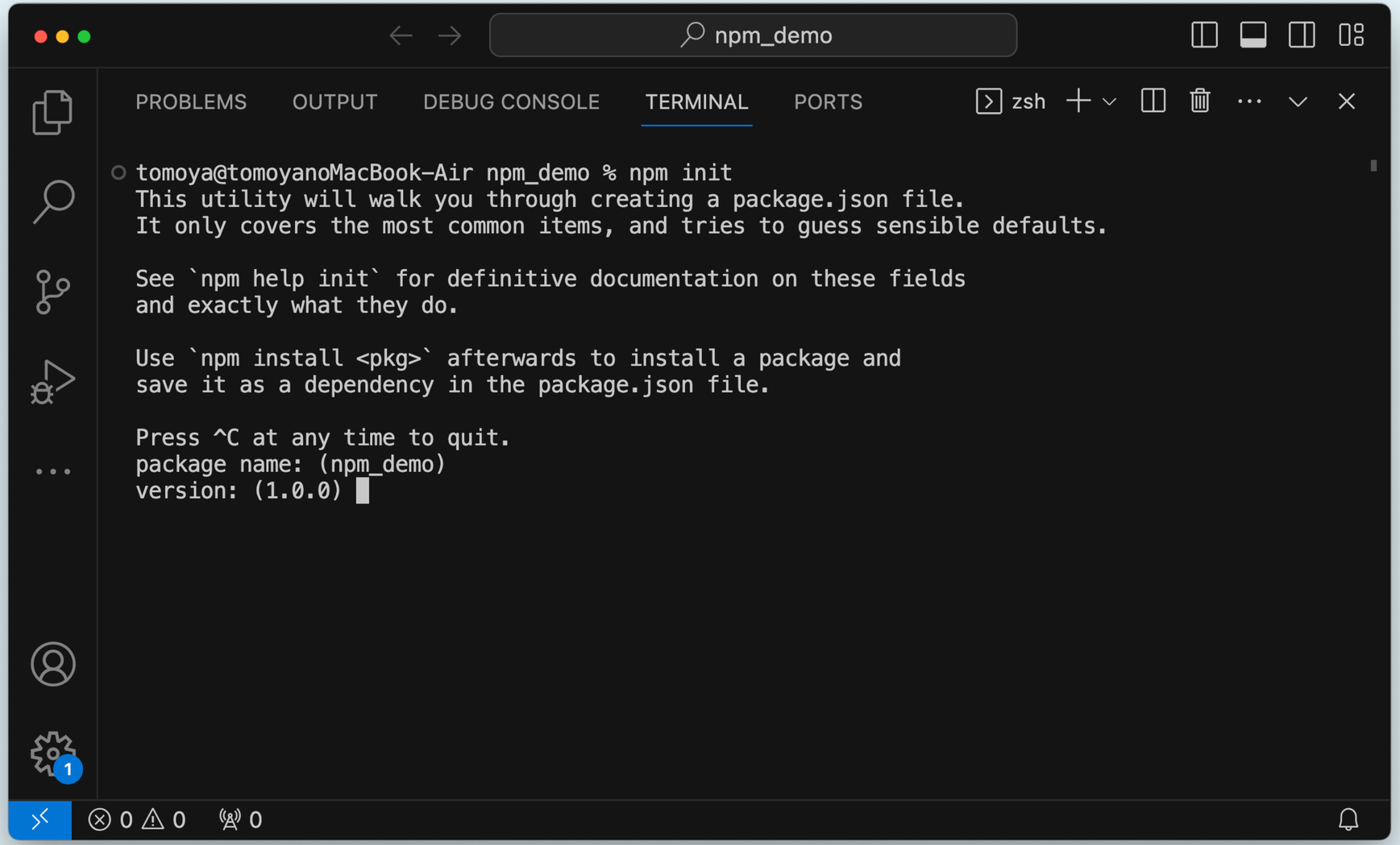


npm init と実行

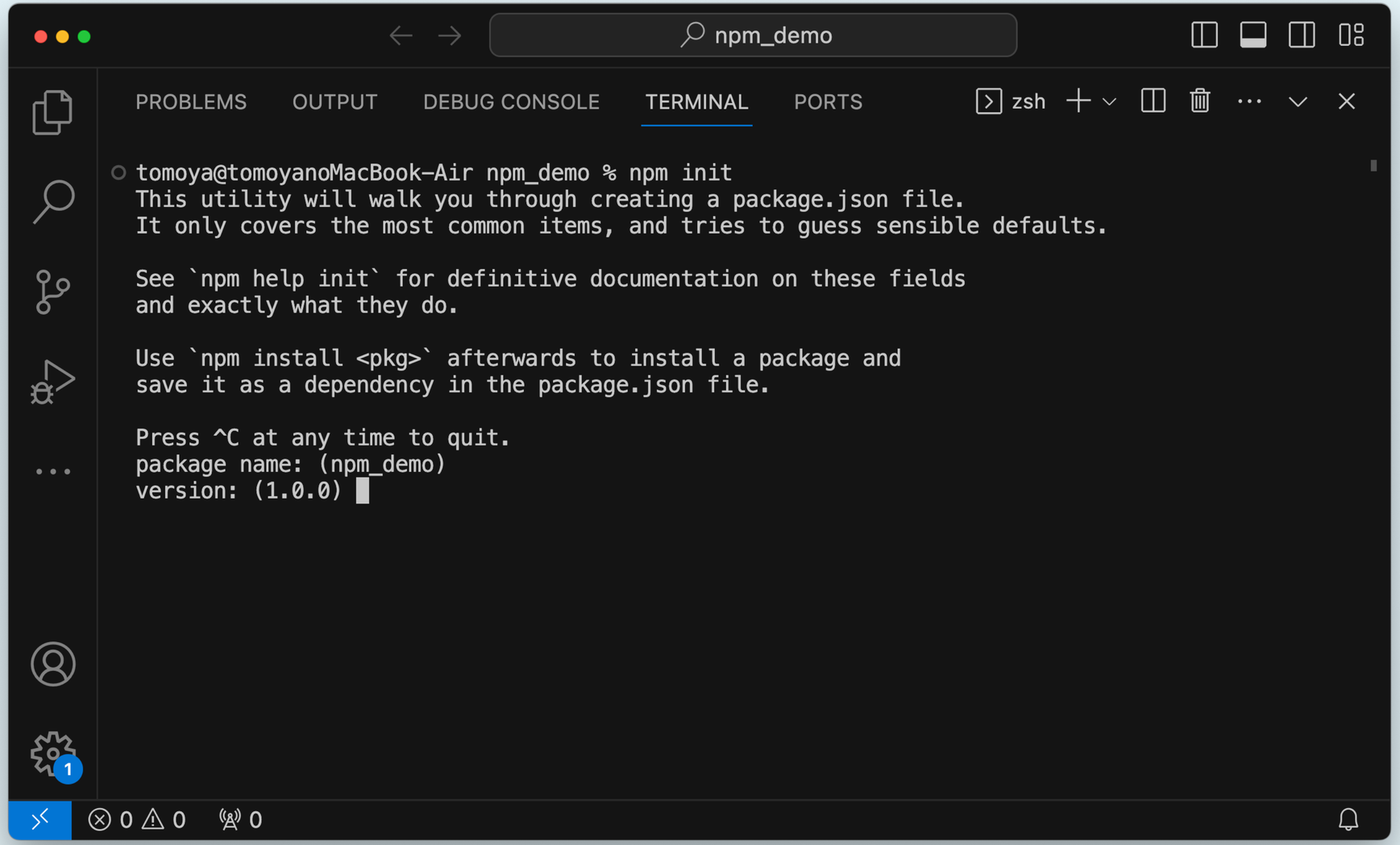
(Node.jsをインストールすればnpmコマンドも使えるはず)



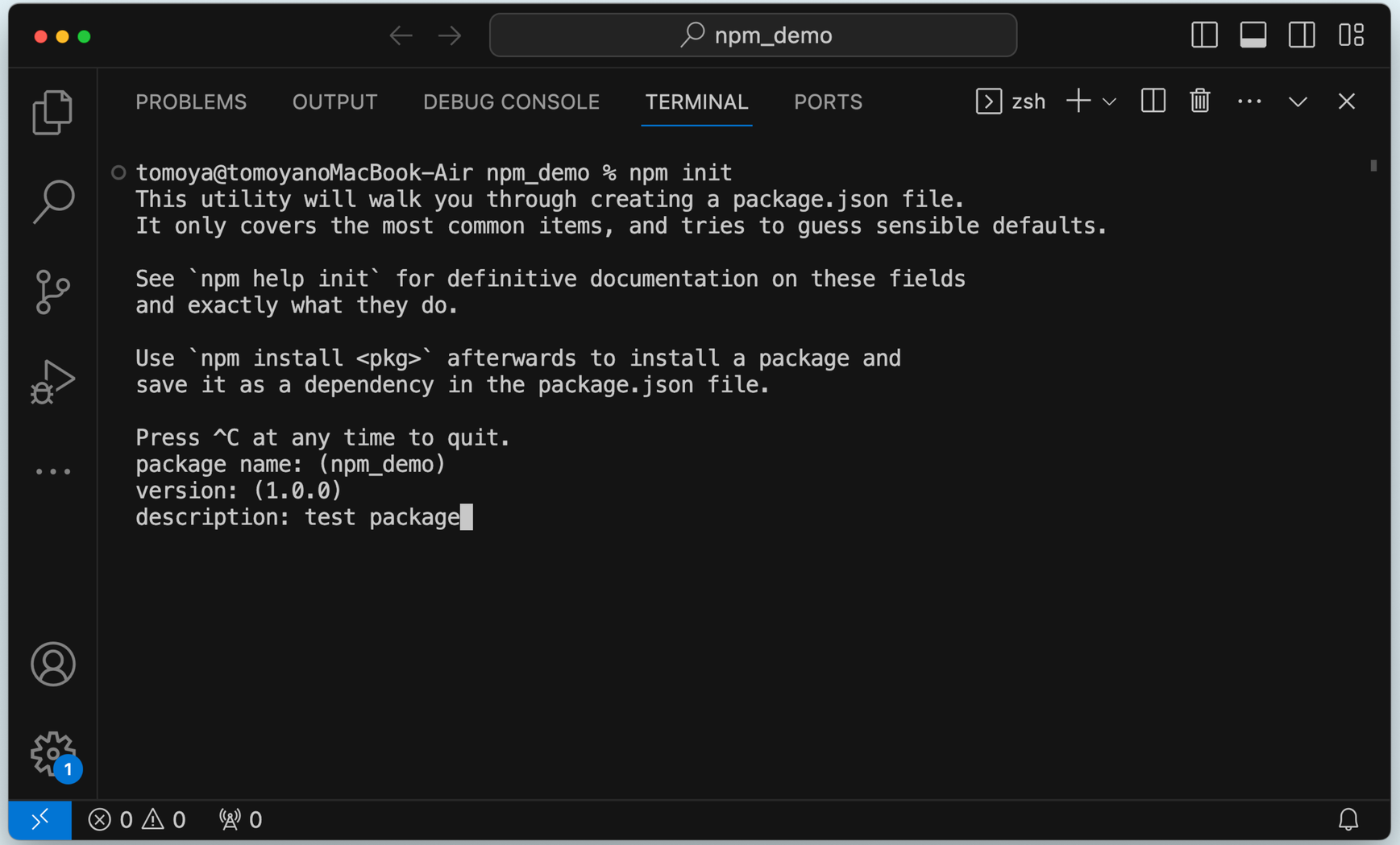
パッケージの名前を聞かれるのでそのままEnter（フォルダ名がパッケージ名になる）



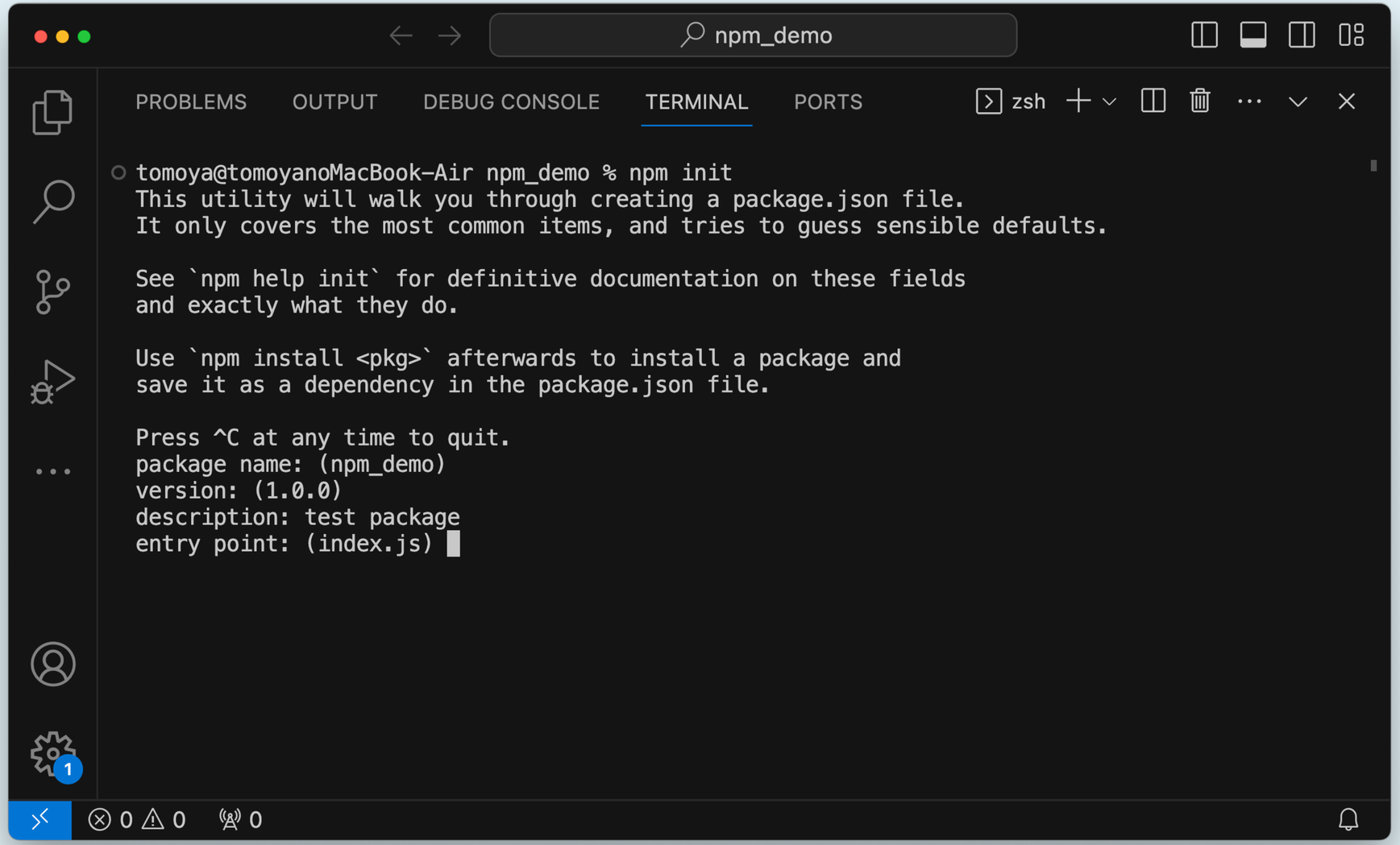
バージョンを聞かれるのでこれも適当にEnter



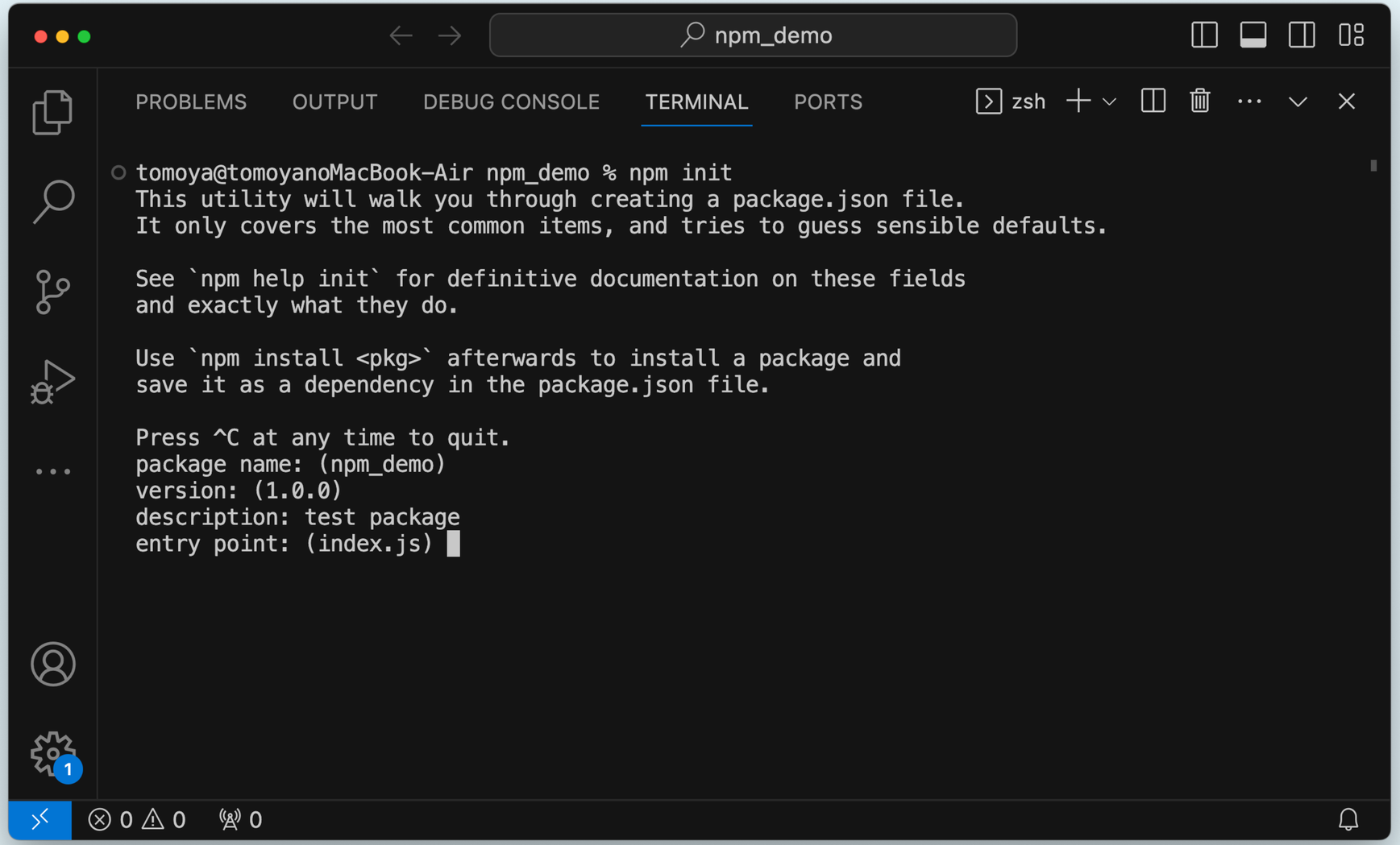
バージョンを聞かれるのでこれも適当にEnter



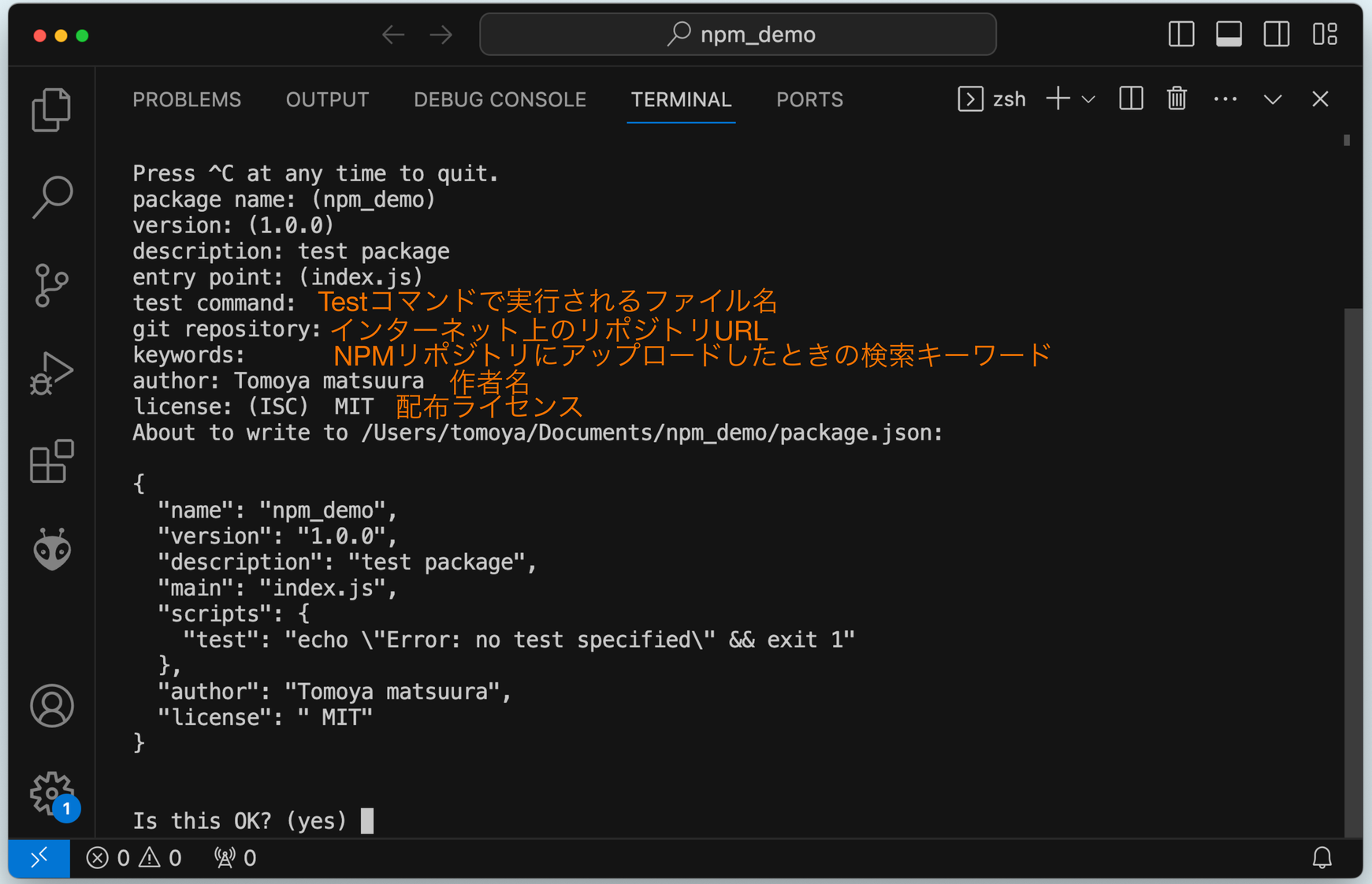
パッケージの説明書きを聞かれるのでEnter

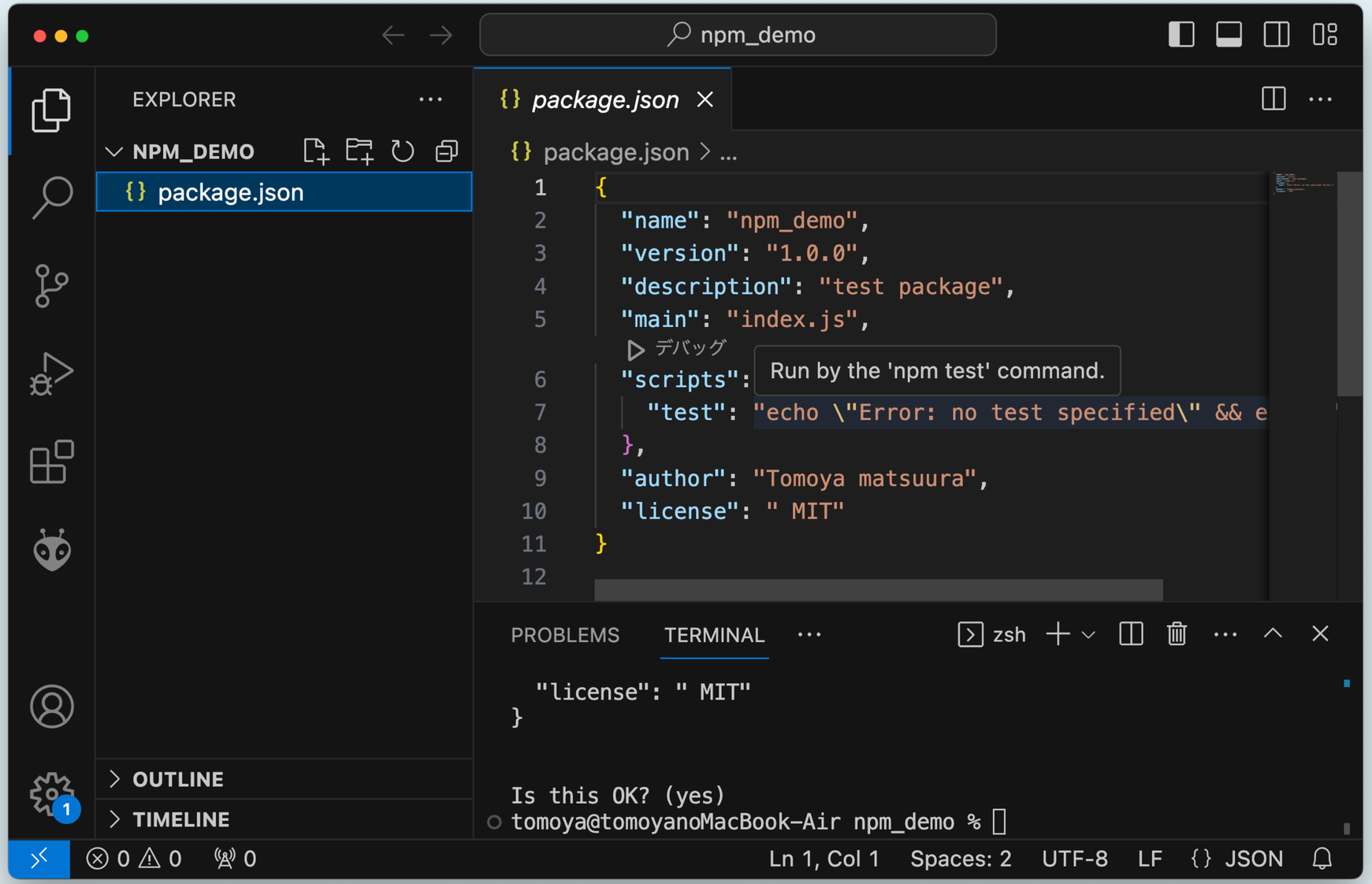


メインのファイル名(npm startコマンドで実行されるもの)を聞かれるのでEnter



メインのファイル名(npm startコマンドで実行されるもの)を聞かれるのでEnter


npm_demo
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
zsh
Press ^C at any time to quit.
package name: (npm_demo)
version: (1.0.0)
description: test package
entry point: (index.js)
test command: Testコマンドで実行されるファイル名
git repository: インターネット上のリポジトリURL
keywords: NPMリポジトリにアップロードしたときの検索キーワード
author: Tomoya matsuura 作者名
license: (ISC) MIT 配布ライセンス
About to write to /Users/tomoya/Documents/npm_demo/package.json:
{
"name": "npm_demo",
"version": "1.0.0",
"description": "test package",
"main": "index.js",
"scripts": {
"test": "echo \"Error: no test specified\" && exit 1"
},
"author": "Tomoya matsuura",
"license": " MIT"
}
Is this OK? (yes)
0 0
0

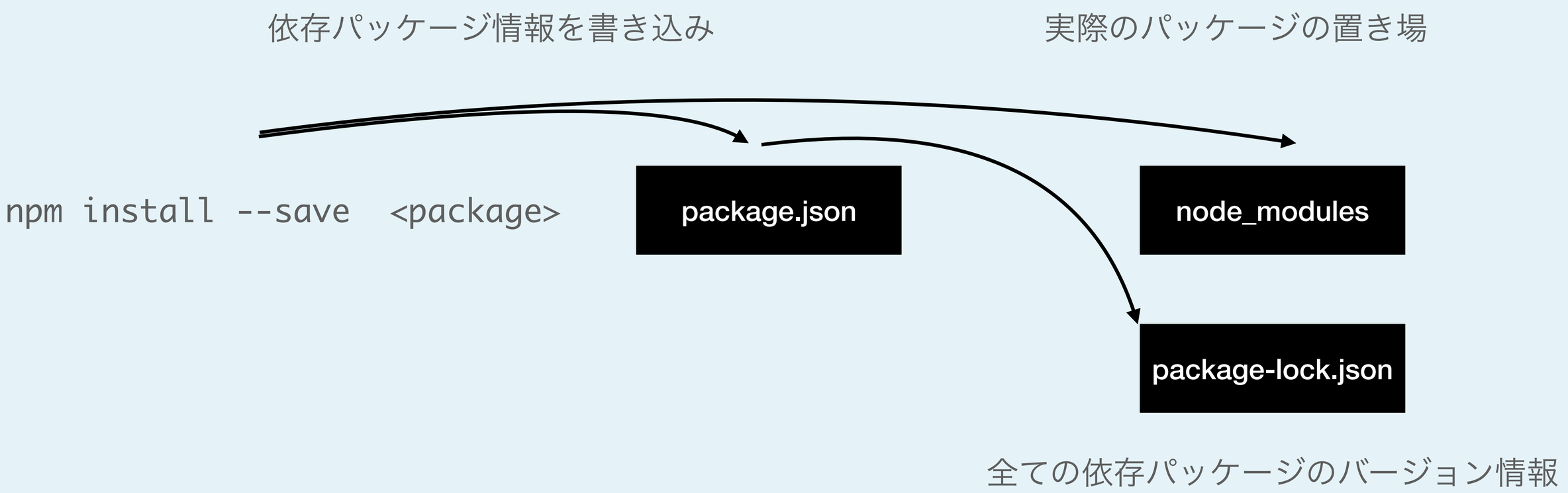


package.jsonというファイルができる

npm init せずに手動でこのファイルを作ってもOK

パッケージを入れてみよう

- 今回使う日本語の自然言語処理ライブラリ、kuromoji.jsを入れてみよう
- npm search kuromojiで調べるとそのまんまkuromojiという名前のパッケージが一番上にヒットする
- npm install --save kuromoji というコマンドを実行
  - --save はpackage.jsonに依存パッケージとしての情報の行を追加してくれる
  - 開発時のみ必要で最終的にはいらないパッケージは--save-devで指定


依存パッケージ情報を書き込み
実際のパッケージの置き場
npm install --save <package>
package.json
node_modules
package-lock.json
全ての依存パッケージのバージョン情報

書き込まれた依存パッケージ情報をもとにインストール

実際のパッケージの置き場

全ての依存パッケージのバージョン情報

package.jsonさえ持ち運べれば必要なパッケージを全て復元できる！

kuromojiで単語を分割してみる

下準備

```
//ファイルシステムの読み込み
const fs = require("fs");
let txt = fs.readFileSync("wagahaiwa_nekodearu.txt").toString();
//余計なルビや注釈とかを正規表現で削除
txt = txt.replace(/《([^《]+)》/gm, "");
txt = txt.replace(/ [#([^ []+)] /gm, "");
fs.writeFileSync("wagahaiwa_nekodearu_plain.txt", txt);
```

単語の分割

```
//kuromojiパッケージの読み込み
const kuromoji = require("kuromoji");
//kuromojiを起動する。内蔵の辞書を読み込む
kuromoji.builder({
    dicPath: "node_modules/kuromoji/dict",
}).build((err, tokenizer) => {
    //この中括弧の中で単語の分割処理をする
    if (err != null) {
        //分割時になんかエラーがあったら終了
        console.error(err);
        return;
    }
    //分割された語の情報はJSONの配列となって帰ってくる
    const tokens = tokenizer.tokenize(txt);
    console.log(tokens)
}
```

オブジェクトデータ（JSON）

- Javascriptでの汎用データフォーマット
- 値はbool、数値、文字列、配列、オブジェクトのどれか
- 配列は数値の配列だけじゃなくオブジェクトの配列や配列の配列など入れ子にできる
- オブジェクトはキーと値のペアの集まりで、値はやはり配列やオブジェクトで入れ子になる
- オブジェクトはjsのコードではobj["key"]かobj.keyのどちらかでアクセスできる

（厳密にはjsコード内のオブジェクトでは値に関数を持つこととかもできるけど、ファイルフォーマットのJSONとしては許されてなかったりと色々違いはある）


```
[{
    "word_id": 594460,
    "word_type": 'KNOWN',
    "word_position": 161,
    "surface_form": '説明',
    "pos": '名詞',
    "pos_detail_1": 'サ変接続',
    "pos_detail_2": '*',
    "pos_detail_3": '*',
    "conjugated_type": '*',
    "conjugated_form": '*',
    "basic_form": '説明',
    "reading": 'セツメイ',
    "pronunciation": 'セツメイ'
  },
  {
    "word_id": 92870,
    "word_type": 'KNOWN',
    "word_position": 163,
    "surface_form": 'や',
    "pos": '助詞',
    "pos_detail_1": '並立助詞',
    "pos_detail_2": '*',
    "pos_detail_3": '*',
    "conjugated_type": '*',
    "conjugated_form": '*',
    "basic_form": 'や',
    "reading": 'ヤ',
    "pronunciation": 'ヤ'
  },
  ...
]
```

オブジェクトデータ（JSON）

- Javascriptでの汎用データフォーマット
- 値はbool、数値、文字列、配列、オブジェクトのどれか
- 配列は数値の配列だけじゃなくオブジェクトの配列や配列の配列など入れ子にできる
- オブジェクトはキーと値のペアの集まりで、値はやはり配列やオブジェクトで入れ子になる
- オブジェクトはjsのコードではobj["key"]かobj.keyのどちらかでアクセスできる

（厳密にはjsコード内のオブジェクトでは値に関数を持つこととかもできるけどファイルフォーマットのJSONとしては許されてなかったりと色々違いはある）


```
[{
    "word_id": 594460,
    "word_type": 'KNOWN',
    "word_position": 161,
    "surface_form": '説明',
    "pos": '名詞',
    "pos_detail_1": 'サ変接続',
    "pos_detail_2": '*',
    "pos_detail_3": '*',
    "conjugated_type": '*',
    "conjugated_form": '*',
    "basic_form": '説明',
    "reading": 'セツメイ',
    "pronunciation": 'セツメイ'
  },
  {
    "word_id": 92870,
    "word_type": 'KNOWN',
    "word_position": 163,
    "surface_form": 'や',
    "pos": '助詞',
    "pos_detail_1": '並立助詞',
    "pos_detail_2": '*',
    "pos_detail_3": '*',
    "conjugated_type": '*',
    "conjugated_form": '*',
    "basic_form": 'や',
    "reading": 'ヤ',
```


このtoken内の情報を利用して
なんやかんやする

```
// わかるものは読みを全てひらがなにして結合
const hiraganas = tokens.map(token => {
    if (token.reading != null) {
        return kanaToHira(token.reading)
    }
    else {
        return token.surface_form
    }
}).join("");
fs.writeFileSync("wagahaiwa_nekodearu_readings.txt", hiraganas);
```

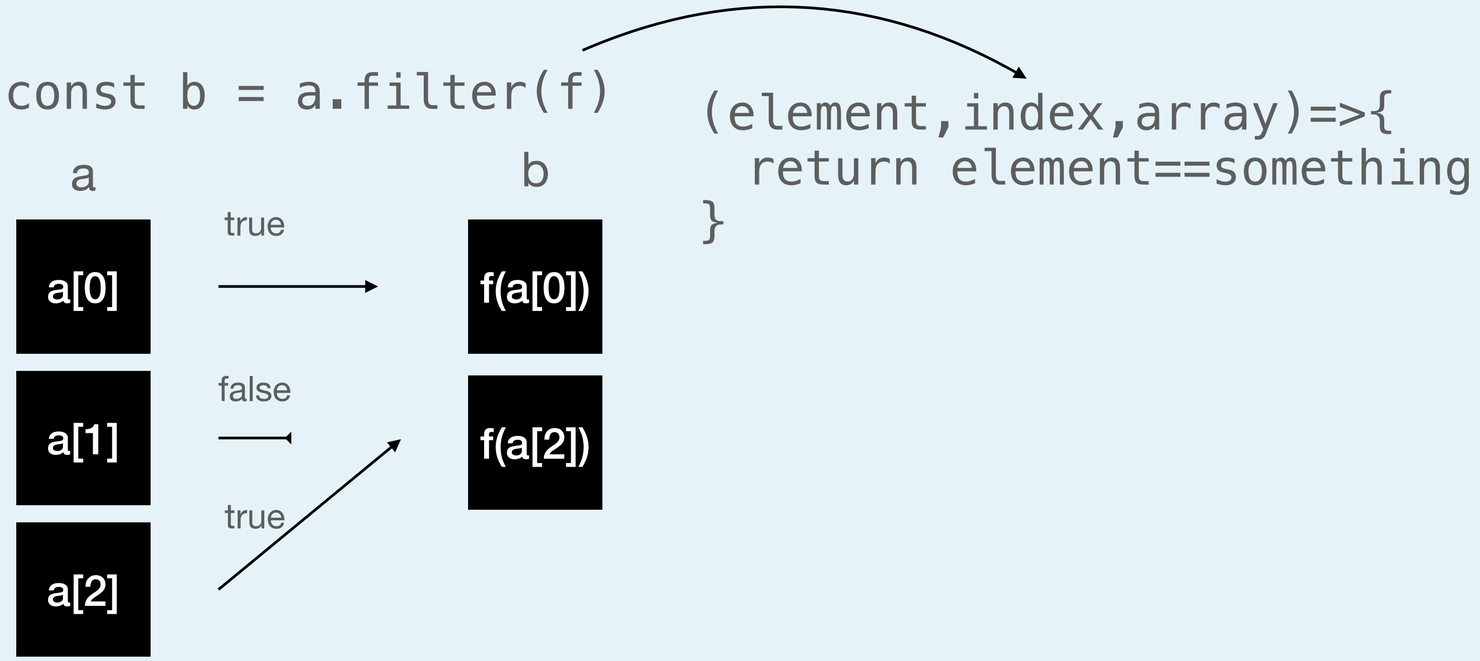
一例：全部ひらがなにしてみる

```
const nouns = tokens.filter((token, idx, tokens) => {
        return token.pos === '名詞' && token.pos_detail_1 === '固有名詞'
    }).map(token => token.surface_form)
        .filter((token, idx, tokens) => tokens.indexOf(token) == idx)//重複を削除
        .sort()//並べ替え
        .join("\n");//改行区切りで文字列として結合
fs.writeFileSync("wagahaiwa_nekodearu_nouns.txt", nouns);
```

一例：固有名詞だけを抜き出してみよう

Array.filter

配列の各要素で当てはまる条件のものを抜き出し



arr.filter(f)を使うと、配列の各要素を使ってfの返り値がtrueのものだけを抜き出せる

単語レベルでのコード・ポエトリー

MUC Loveletter Generator

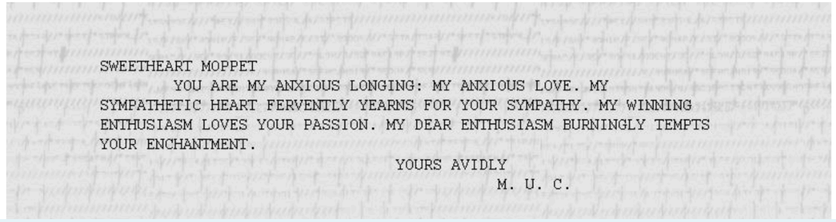
Christopher Strachey, Alan Turing

A Queer History of Computing: Part Three

by Jacob Gaboury
2013-04-09

For the preceding segment of this four part geneology, see Part 2

In this third segment of our genealogy we begin to form a connection, and to examine those lesser-known but foundational figures that radiate out from Turing's early work. Perhaps appropriately, given the venue, this second figure leads us to one of the earliest examples of computational art ever produced, though he did not claim the title of artist for himself. This history also moves us forward to those pivotal years surrounding Turing's arrest and death. While Turing underwent a highly visible crisis, Christopher Strachey's work was coming into its own. Once again the connection is tenuous, and little record survives to document more than a passing relationship between these two men, but what remains is a surprisingly poetic attempt to play at the machine.

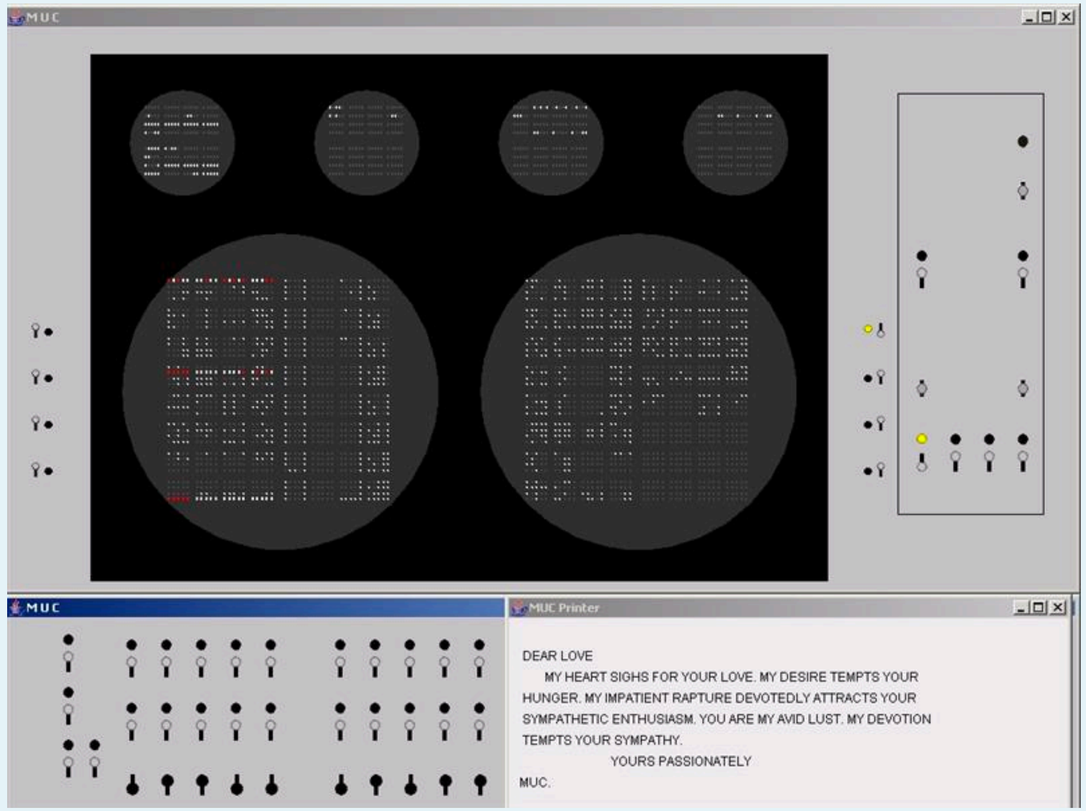


SWEETHEART MOPPET
YOU ARE MY ANXIOUS LONGING: MY ANXIOUS LOVE. MY
SYMPATHETIC HEART FERVENTLY YEARNS FOR YOUR SYMPATHY. MY WINNING
ENTHUSIASM LOVES YOUR PASSION. MY DEAR ENTHUSIASM BURNINGLY TEMPTS
YOUR ENCHANTMENT.
YOURS AVIDLY
M. U. C.

https://rhizome.org/editorial/2013/apr/9/queer-history-computing-part-three/

MUC Loveletter Generator

Christopher Strachey, Alan Turing



David Linkによるエミュレーション

https://web.archive.org/web/20130216222649/http://alpha60.de/research/muc/

MUC Loveletter Generator

Christopher Strachey, Alan Turing

CHRISTOPHER STRACHEY "LOVELETTERS" (1952)

BELOVED DUCK,
YOU ARE MY PASSIONATE FANCY, MY UNSATISFIED ENTHUSIASM. MY ADORATION SEDUCTIVELY ADORES YOUR BURNING. MY INFATUATION ADORES YOUR SWEET CHARM. YOU ARE MY BEAUTIFUL FONDNESS.
YOURS ARDENTLY,
M.U.C.

GENERATE ANOTHER LOVELETTER

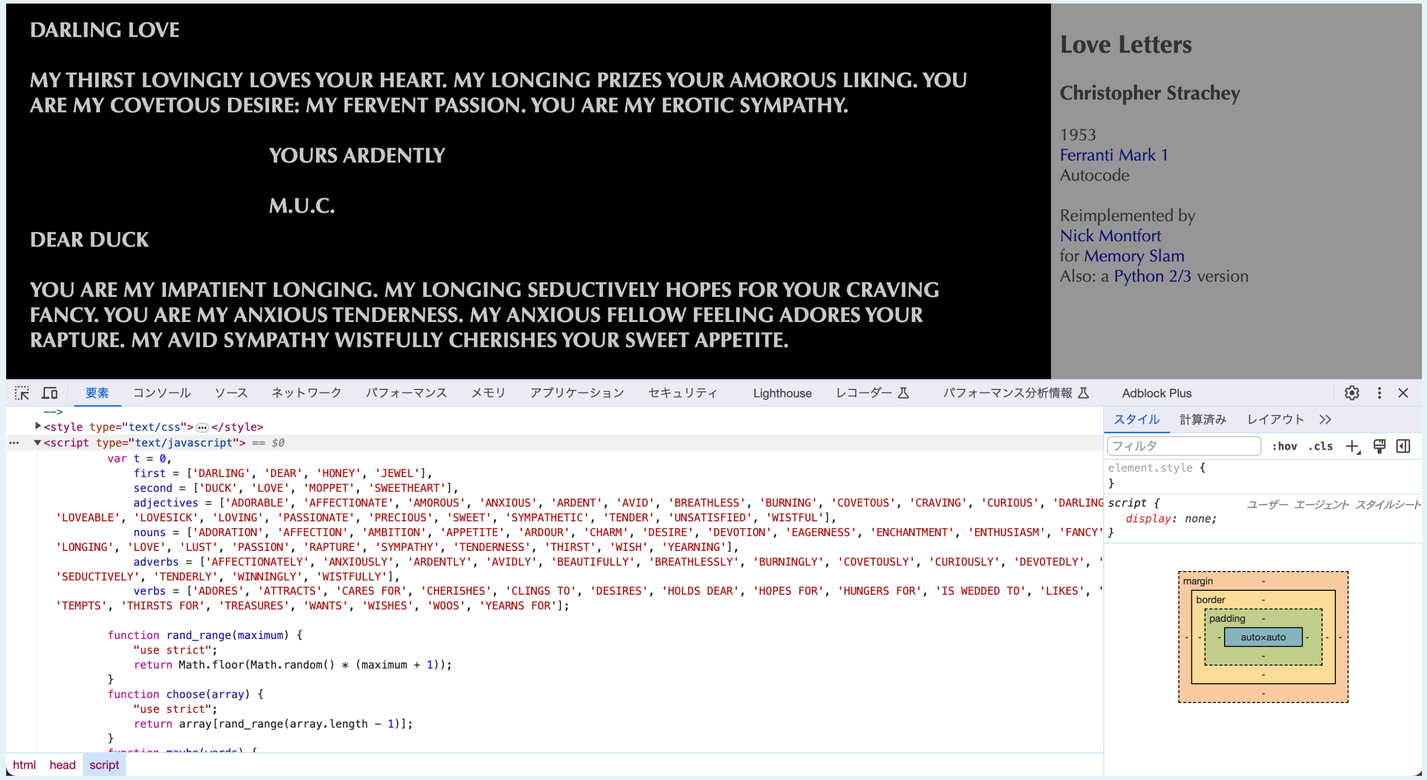
- SALUTATIONS1
  - BELOVED
  - DARLING
  - DEAR
  - DEAREST
  - FANCIFUL
  - HONEY
- SALUTATIONS2
  - CHICKPEA
  - DEAR
  - DUCK
  - JEWEL
  - LOVE
  - MOPPET
  - SWEETHEART
- ADJECTIVES
  - AFFECTIONATE
  - AMOROUS
  - ANXIOUS
  - AVID
  - BEAUTIFUL
  - BREATHLESS
  - BURNING
  - COVETOUS
  - CRAVING
  - CURIOUS
  - EAGER
  - FERVENT
  - FONDEST
  - LOVEABLE
  - LOVESICK
  - LOVING
  - PASSIONATE
  - PRECIOUS
  - SEDUCTIVE
  - SWEET
  - SYMPATHETIC
  - TENDER
- NOUNS
  - ADORATION
  - AFFECTION
  - AMBITION
  - APPETITE
  - ARDOUR
  - BEING
  - BURNING
  - CHARM
  - CRAVING
  - DESIRE
  - DEVOTION
  - EAGERNESS
  - ENCHANTMENT
  - ENTHUSIASM
  - FANCY
  - FELLOW FEELING
  - FERVOUR
  - FONDNESS
  - HEART
  - HUNGER
  - INFATUATION
  - LITTLE LIKING
- ADVERBS
  - AFFECTIONATELY
  - ARDENTLY
  - ANXIOUSLY
  - BEAUTIFULLY
  - BURNINGLY
  - COVETOUSLY
  - CURIOUSLY
  - EAGERLY
  - FERVENTLY
  - FONDLY
  - IMPATIENTLY
  - KEENLY
  - LOVINGLY
  - PASSIONATELY
  - SEDUCTIVELY
  - TENDERLY
  - WISTFULLY
- VERBS
  - ADORES
  - ATTRACTS
  - CLINGS TO
  - HOLDS DEAR
  - HOPES FOR
  - HUNGERS FOR
  - LIKES
  - LONGS FOR
  - LOVES
  - LUSTS AFTER
  - PANTS FOR
  - PINES FOR
  - SIGHS FOR
  - TEMPTS
  - THIRSTS FOR
  - TREASURES
  - YEARNS FOR
  - WOOS

Matt SephtonによるPHPでの実装

https://www.gingerbeardman.com/loveletter/

MUC Loveletter Generator

Christopher Strachey, Alan Turing



Nick Montfortによるブラウザ上Javascriptでの実装

https://nickm.com/memslam/love_letters.html

https://nickm.com/memslam/love_letters.py　Python版も

偶然短歌bot（いなにわ）

https://twitter.com/g57577

偶然短歌bot（いなにわ）

guuzen-tanka / pickup.rb ↑ Top

Code Blame Raw

```ruby
def check_tanka(tanka, next_node)
  # 括弧
  parentheses = [0] * PARENTHESES_LEFT.length
  tanka.each do |node|
    idx = PARENTHESES_LEFT.index(node[:surface])
    unless idx.nil?
      parentheses[idx] += 1
    end
    idx = PARENTHESES_RIGHT.index(node[:surface])
    unless idx.nil?
      parentheses[idx] -= 1
      return false if parentheses[idx] < 0
    end
  end
  return false unless parentheses == [0] * PARENTHESES_LEFT.length
  # 末尾
  return false if "連体詞" == tanka[-1][:feature]["品詞"]
  return false if /^(名詞接続|格助詞|係助詞|連体化|接続助詞|並立助詞|副詞化|数接続)$/ =~
                  tanka[-1][:feature]["品詞細分類1"]
  return false if "助動詞" == tanka[-1][:feature]["品詞"] and
                  "だ" == tanka[-1][:feature]["原形"]
  return false if !next_node.nil? and
                  !check_first_word(next_node, COUNTS.length)
  # OK
  return true
end

def find_tanka(texts)
  tankas = []
  texts.each do |text|
    nm_nodes = $nm.parse_as_nodes(text)
    nodes = convert_nodes(nm_nodes)
    nl = nodes.length
    (0...nl).each do |i|
```

https://github.com/inaniwa3/guuzen-tanka