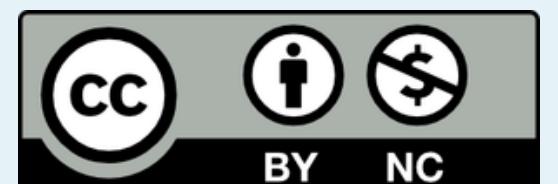


# メディアアート・プログラミング2

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](mailto:matsura.tomoya@noc.geidai.ac.jp) [teach@matsuuratomo.ya.com](mailto:teach@matsuuratomo.ya.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

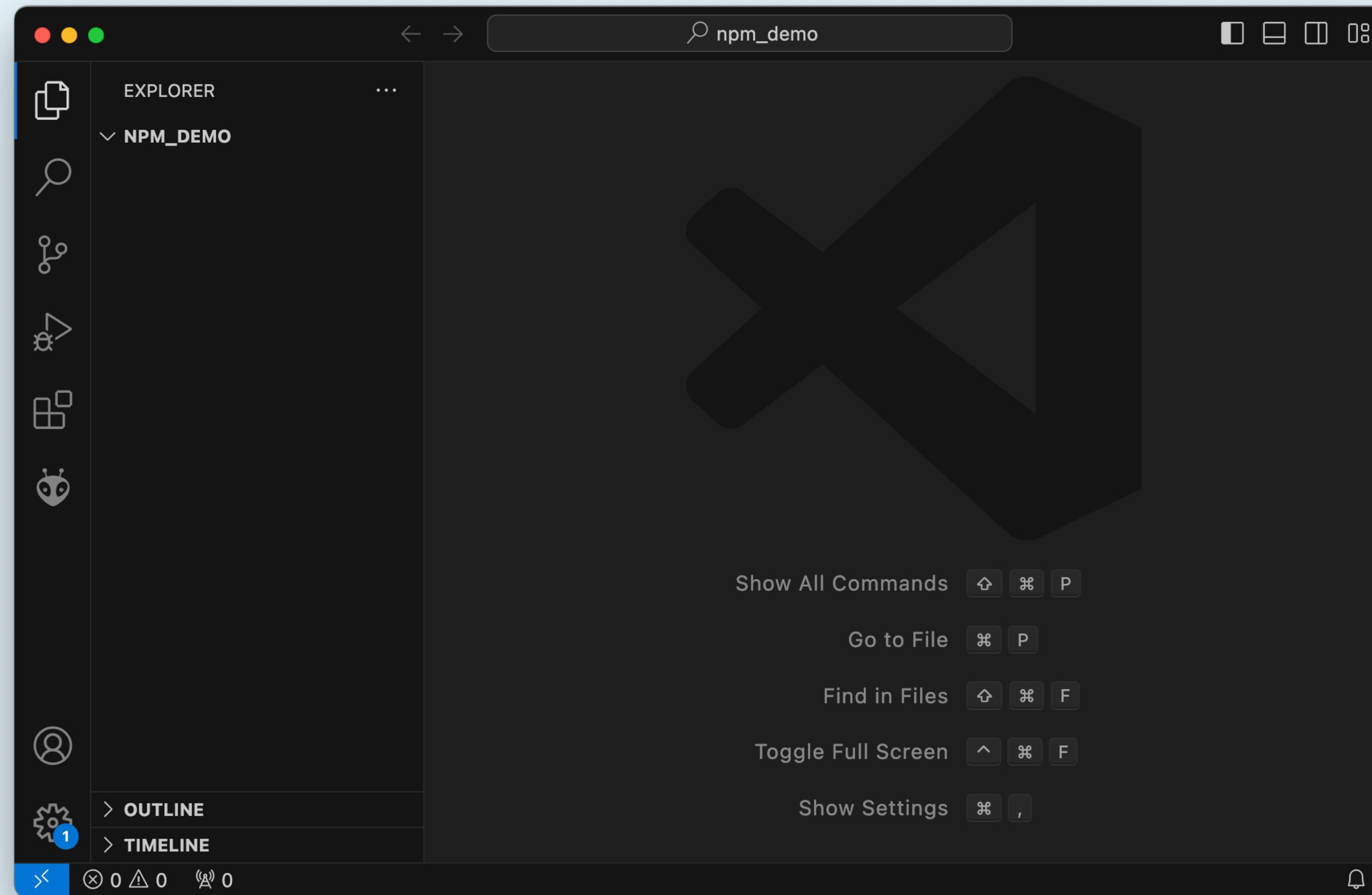
\*Javascriptではライブラリという言葉を使うことはあまりなく、  
言語ごとに使い分けも微妙に異なるのでざっくりした一般論くらいに思ってください

# 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用可能な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が多い  
(Node.jsはnpm、Pythonはpipなど)
- パッケージ：ライブラリやモジュールなどをまとめたもの
- 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

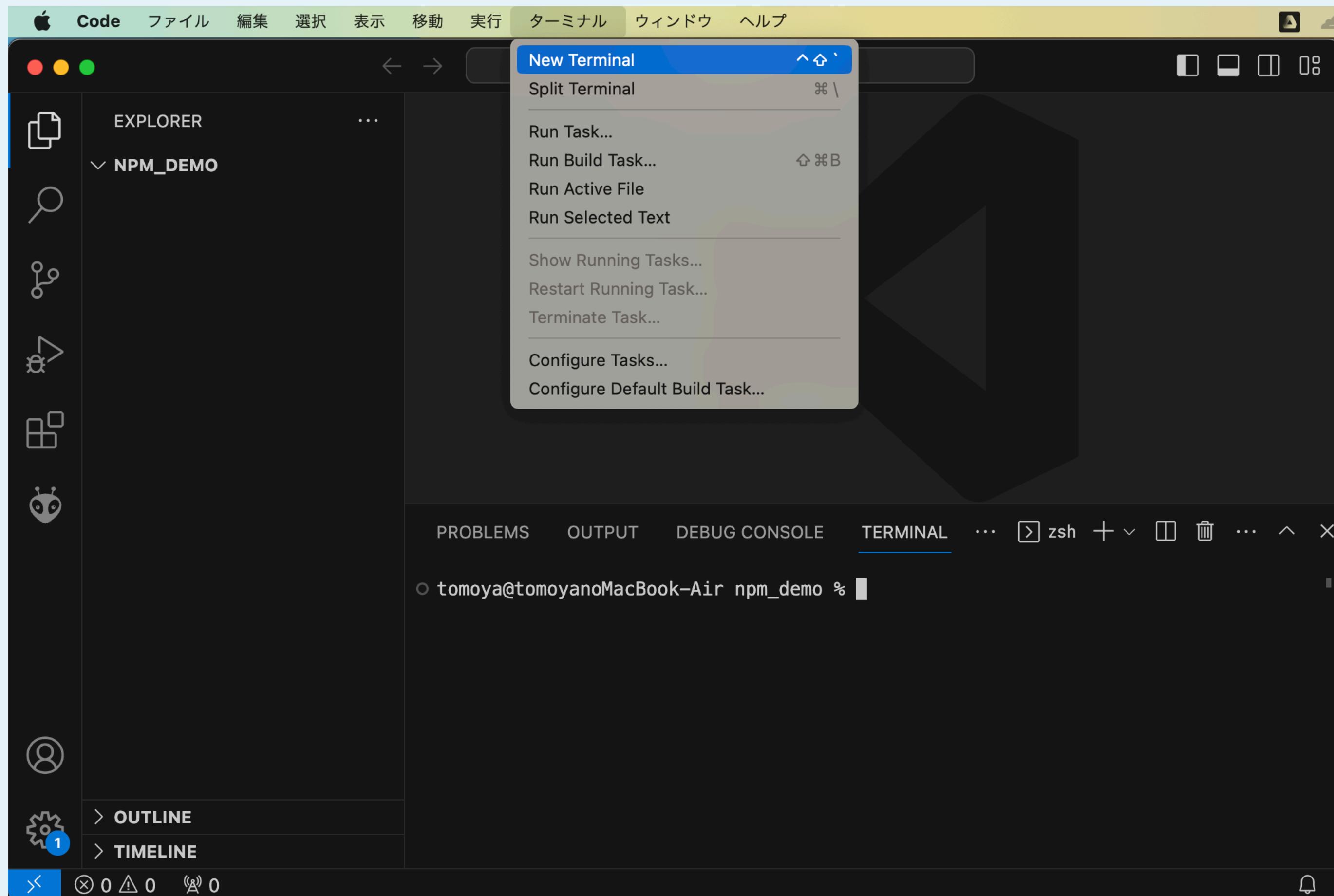
\*Javascriptではライブラリという言葉を使うことはあまりなく、  
言語ごとに使い分けも微妙に異なるのでざっくりした一般論くらいに思ってください

# NPM(node package manager)を使ってみよう



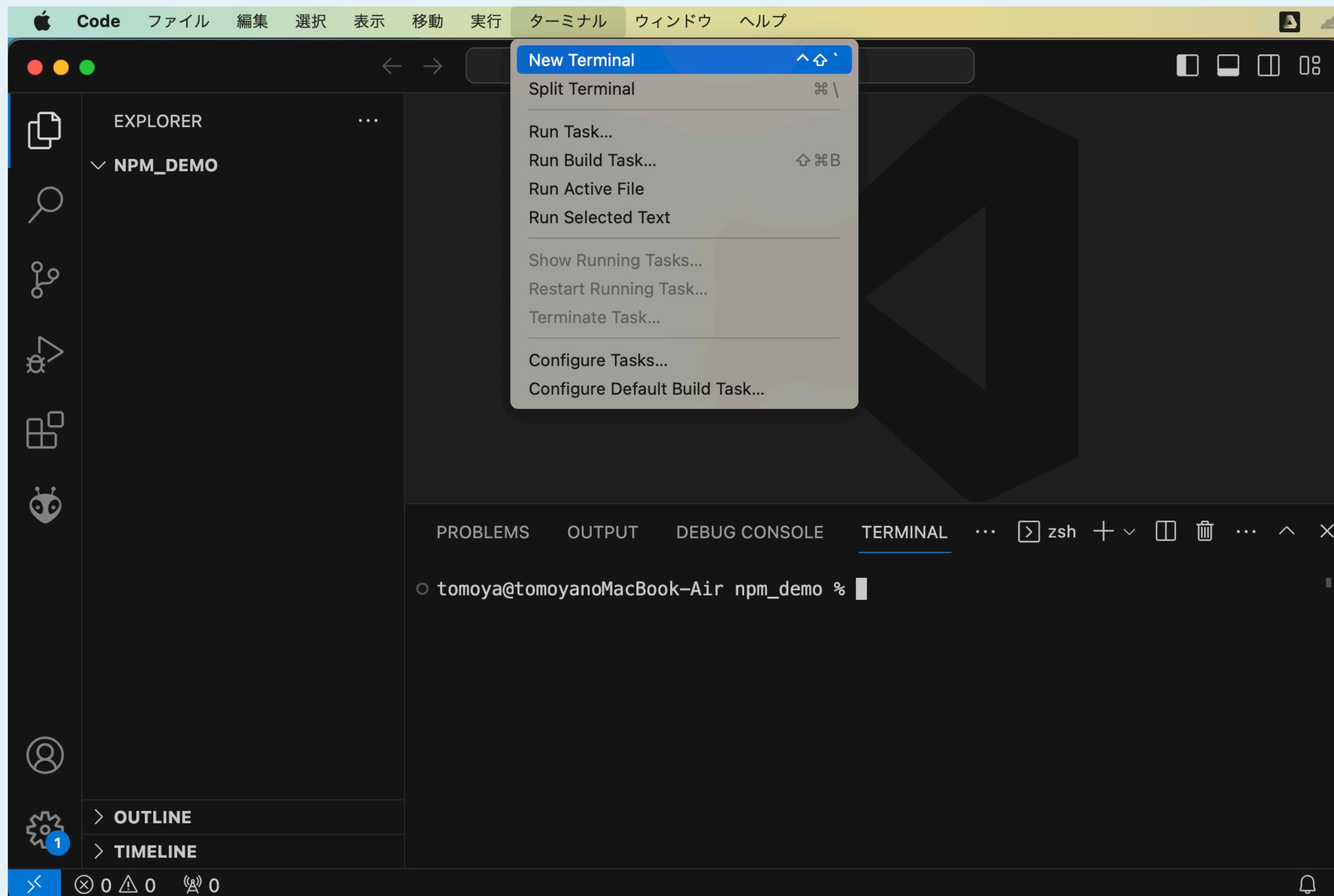
空のフォルダを作ってVSCodeで開く

# NPM(node package manager)を使ってみよう



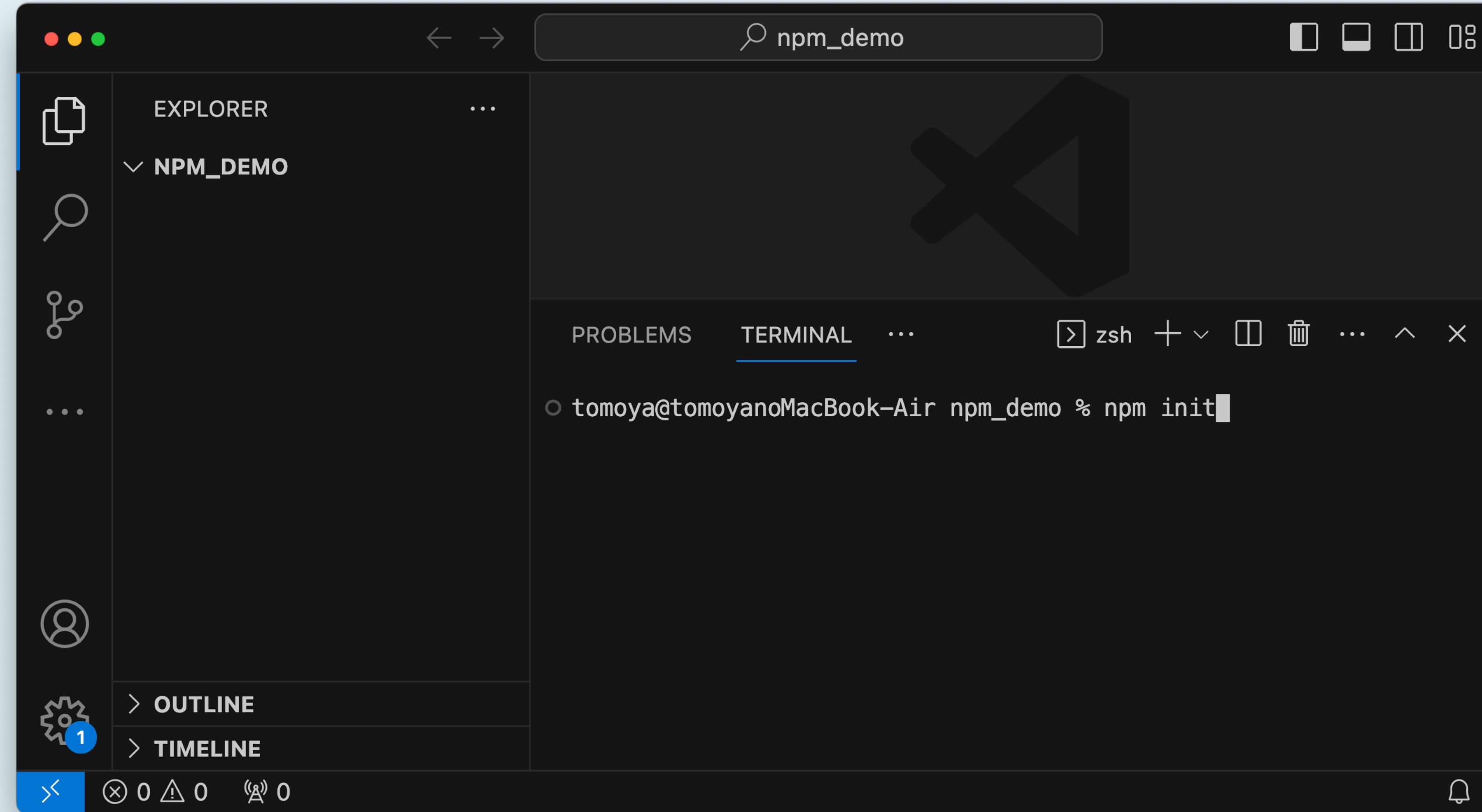
ターミナルを開く

# 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)



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

zsh + ⌂ ⌂ ⌂ ⌂ ⌂ ⌂

tomoya@tomoyanoMacBook-Air npm\_demo % npm init  
This utility will walk you through creating a package.json file.  
It only covers the most common items, and tries to guess sensible defaults.  
See `npm help init` for definitive documentation on these fields  
and exactly what they do.  
Use `npm install <pkg>` afterwards to install a package and  
save it as a dependency in the package.json file.  
Press ^C at any time to quit.  
package name: (npm\_demo) █

色々聞かれるけど後から修正できるので  
Enter連打でも大丈夫

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）

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS > zsh + ↻

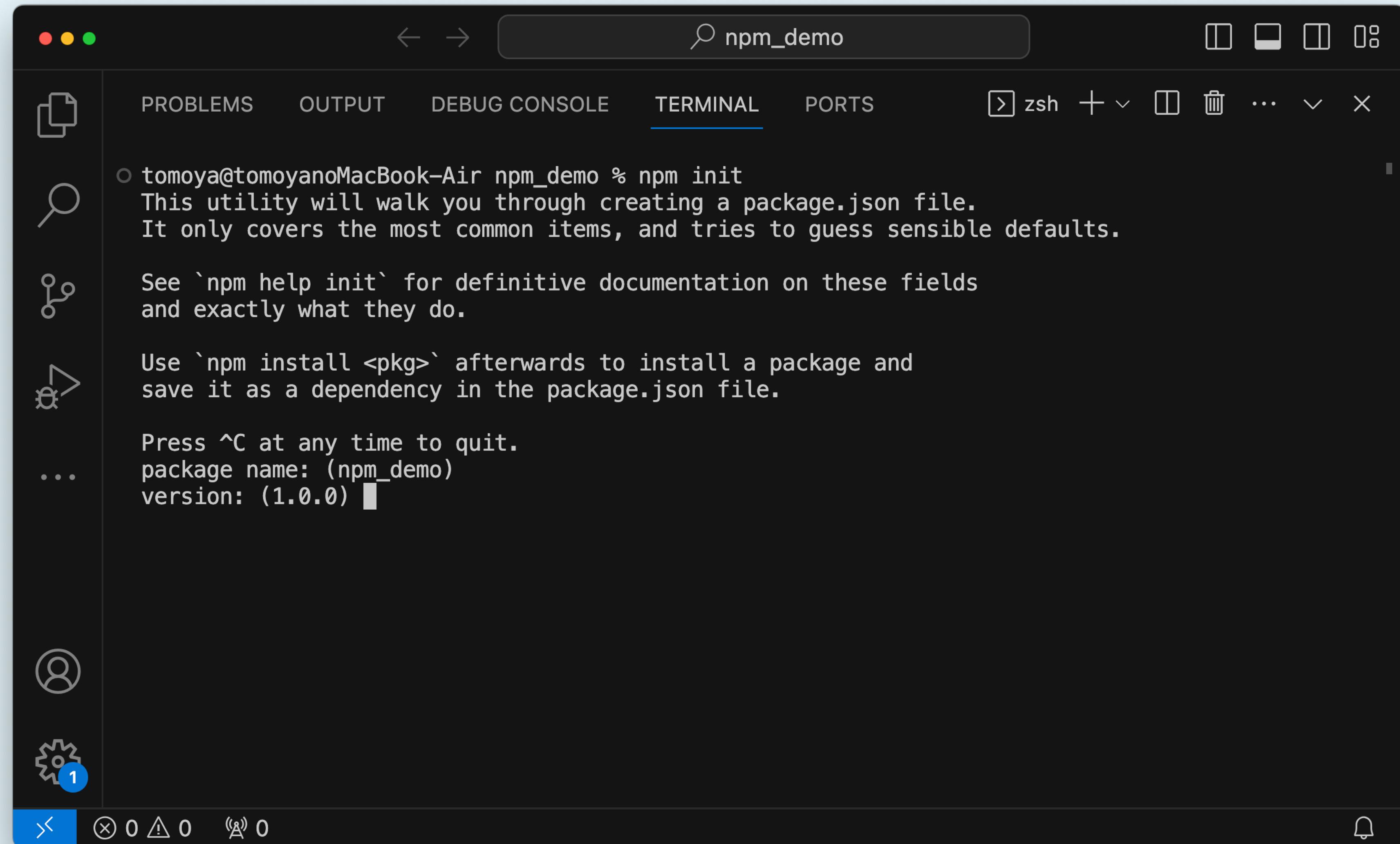
```
tomoya@tomoyanoMacBook-Air npm_demo % npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (npm_demo)
version: (1.0.0) █
```

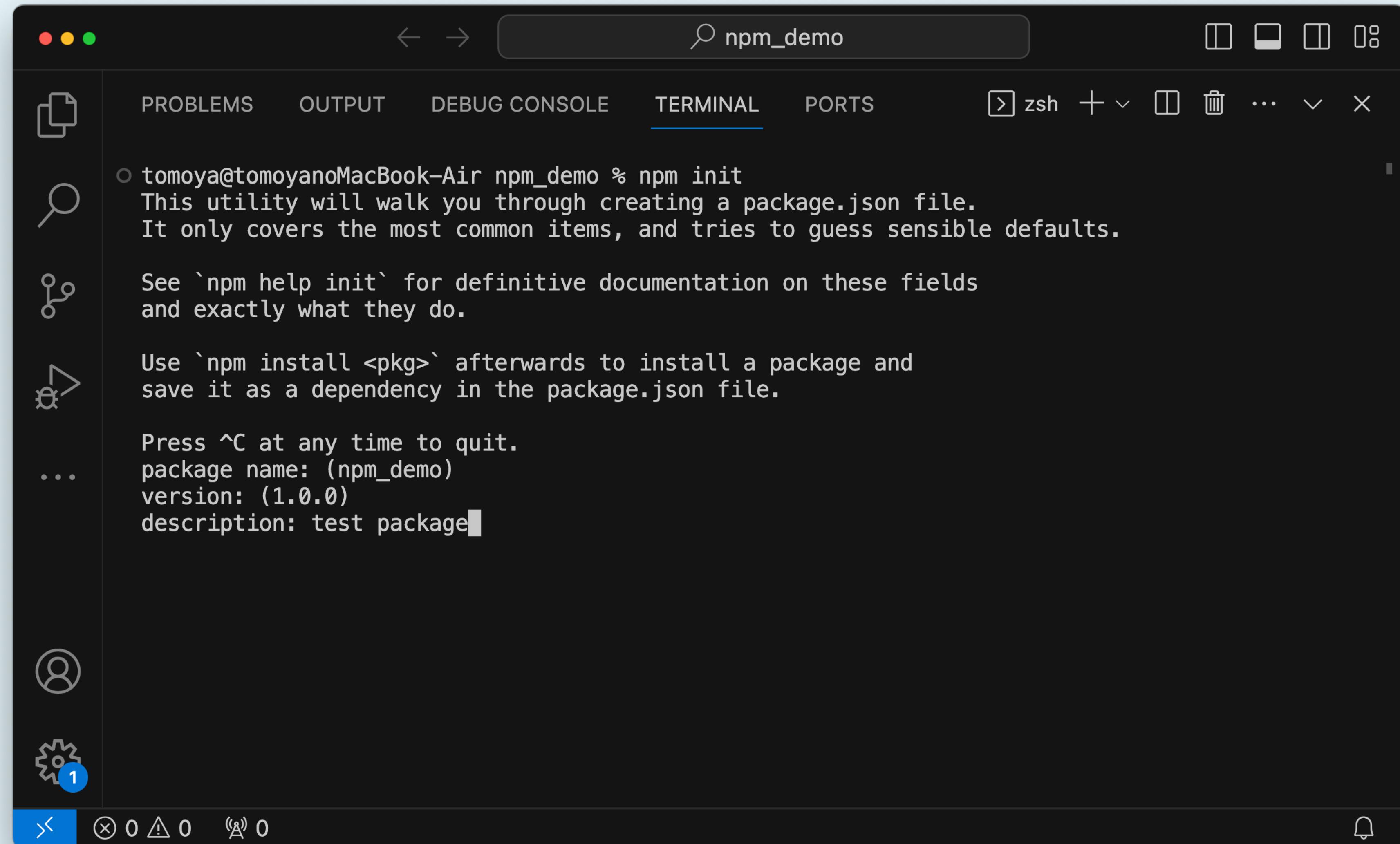
バージョンを聞かれるのでこれも適当にEnter



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

tomoya@tomoyanoMacBook-Air npm\_demo % npm init  
This utility will walk you through creating a package.json file.  
It only covers the most common items, and tries to guess sensible defaults.  
See `npm help init` for definitive documentation on these fields  
and exactly what they do.  
Use `npm install <pkg>` afterwards to install a package and  
save it as a dependency in the package.json file.  
Press ^C at any time to quit.  
package name: (npm\_demo)  
version: (1.0.0) █

バージョンを聞かれるのでこれも適当にEnter



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

tomoya@tomoyanoMacBook-Air npm\_demo % npm init

This utility will walk you through creating a package.json file.  
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields  
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and  
save it as a dependency in the package.json file.

Press ^C at any time to quit.

package name: (npm\_demo)  
version: (1.0.0)  
description: test package█

パッケージの説明書きを聞かれるのでEnter

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

tomoya@tomoyanoMacBook-Air npm\_demo % npm init  
This utility will walk you through creating a package.json file.  
It only covers the most common items, and tries to guess sensible defaults.  
See `npm help init` for definitive documentation on these fields  
and exactly what they do.  
Use `npm install <pkg>` afterwards to install a package and  
save it as a dependency in the package.json file.  
Press ^C at any time to quit.  
package name: (npm\_demo)  
version: (1.0.0)  
description: test package  
entry point: (index.js)

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

tomoya@tomoyanoMacBook-Air npm\_demo % npm init  
This utility will walk you through creating a package.json file.  
It only covers the most common items, and tries to guess sensible defaults.  
See `npm help init` for definitive documentation on these fields  
and exactly what they do.  
Use `npm install <pkg>` afterwards to install a package and  
save it as a dependency in the package.json file.  
Press ^C at any time to quit.  
package name: (npm\_demo)  
version: (1.0.0)  
description: test package  
entry point: (index.js)

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

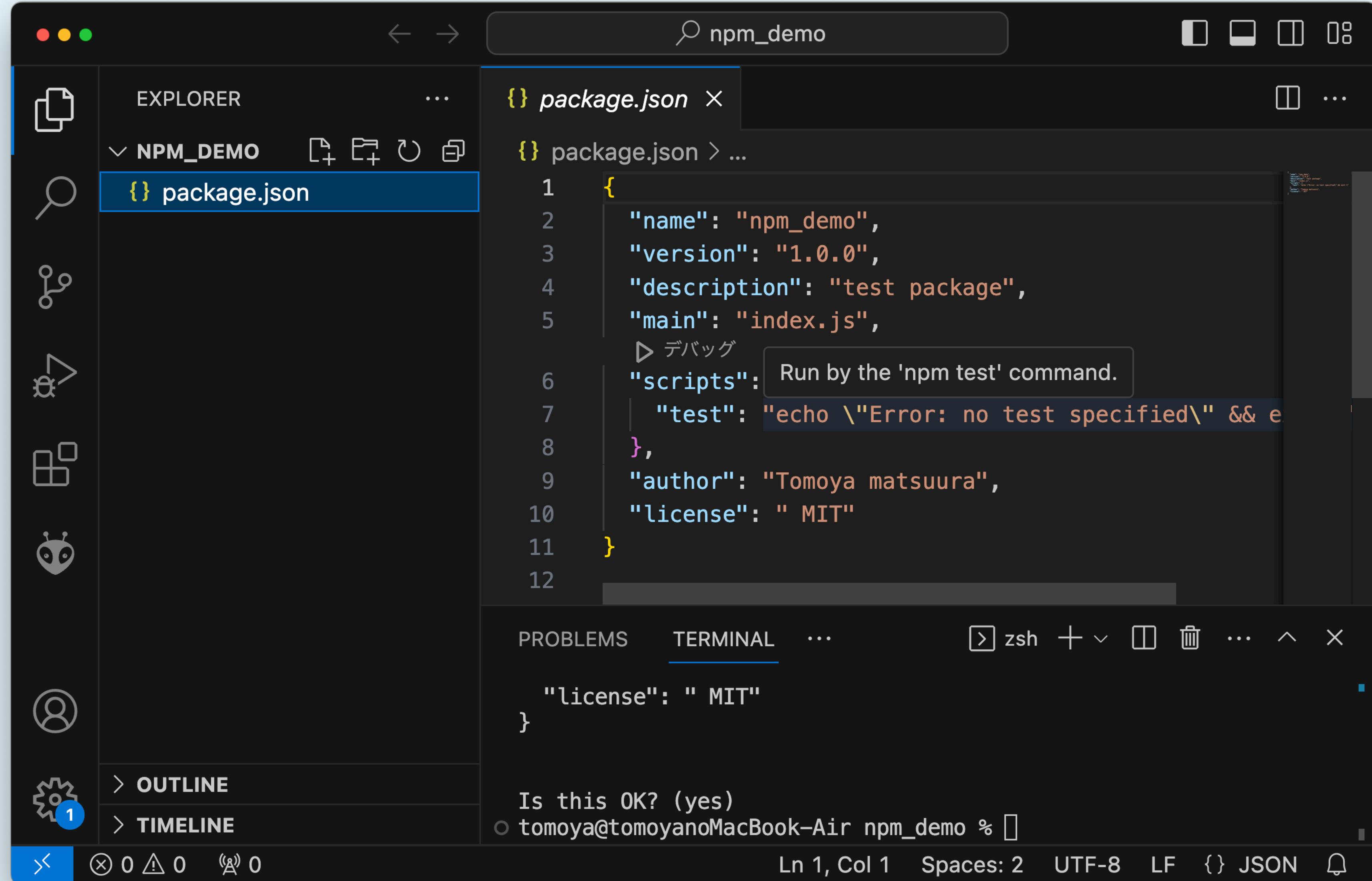
A screenshot of a macOS terminal window titled "npm\_demo". The window has a dark theme. On the left is a vertical toolbar with icons for file operations, search, git, terminal, and settings. The main area shows the command-line interface for creating an npm package named "npm\_demo". The package details listed are:

```
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) █
```

The bottom status bar shows the number of files (1), and the bottom right corner has a bell icon.



The screenshot shows the Visual Studio Code interface with a dark theme. The Explorer sidebar on the left has a tree view with a single item: 'NPM\_DEMO' expanded, showing 'package.json'. The main editor area displays the contents of 'package.json':

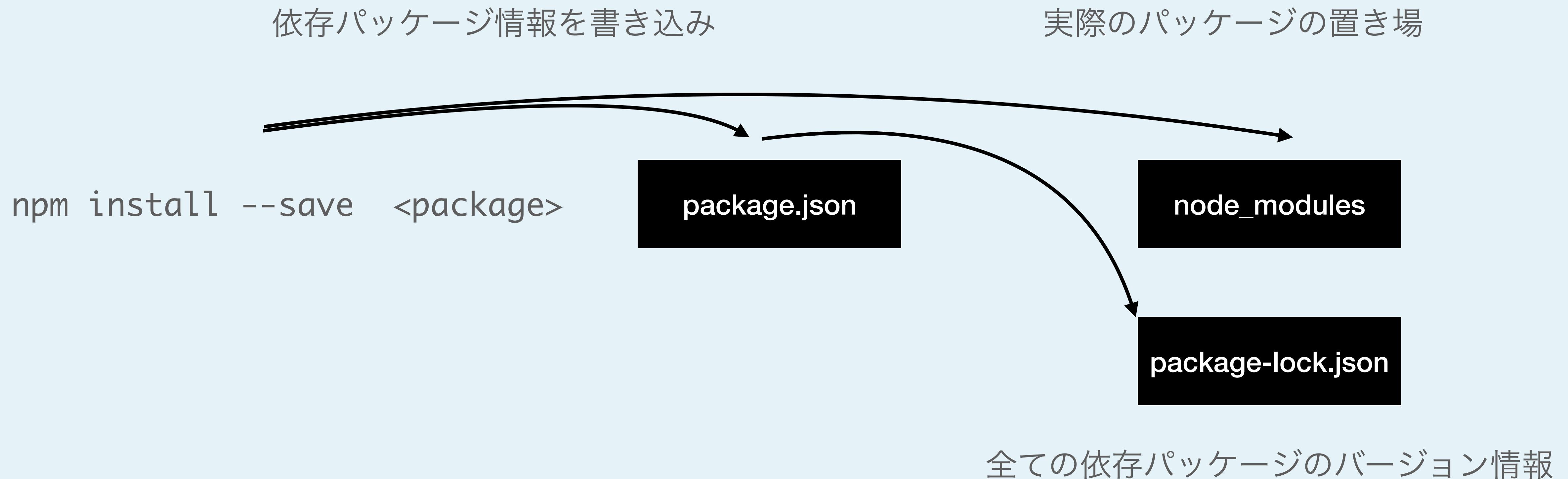
```
1 {  
2   "name": "npm_demo",  
3   "version": "1.0.0",  
4   "description": "test package",  
5   "main": "index.js",  
6   "scripts": "Run by the 'npm test' command.  
7     "test": "echo \\\"Error: no test specified\\\" && e  
8   },  
9   "author": "Tomoya matsuura",  
10  "license": " MIT"  
11}  
12
```

The status bar at the bottom indicates the file is JSON, has 1 line and 2 spaces, and is in UTF-8 encoding.

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

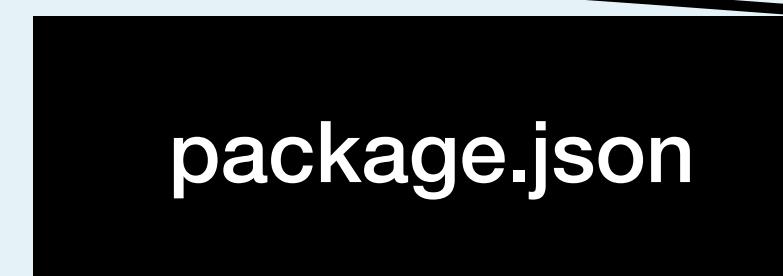
# パッケージを入れてみよう

- ・ 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



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

npm install



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つたりと色々違うはある)

```
[{"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うはある)

```
[{"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"}]
```

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);
```

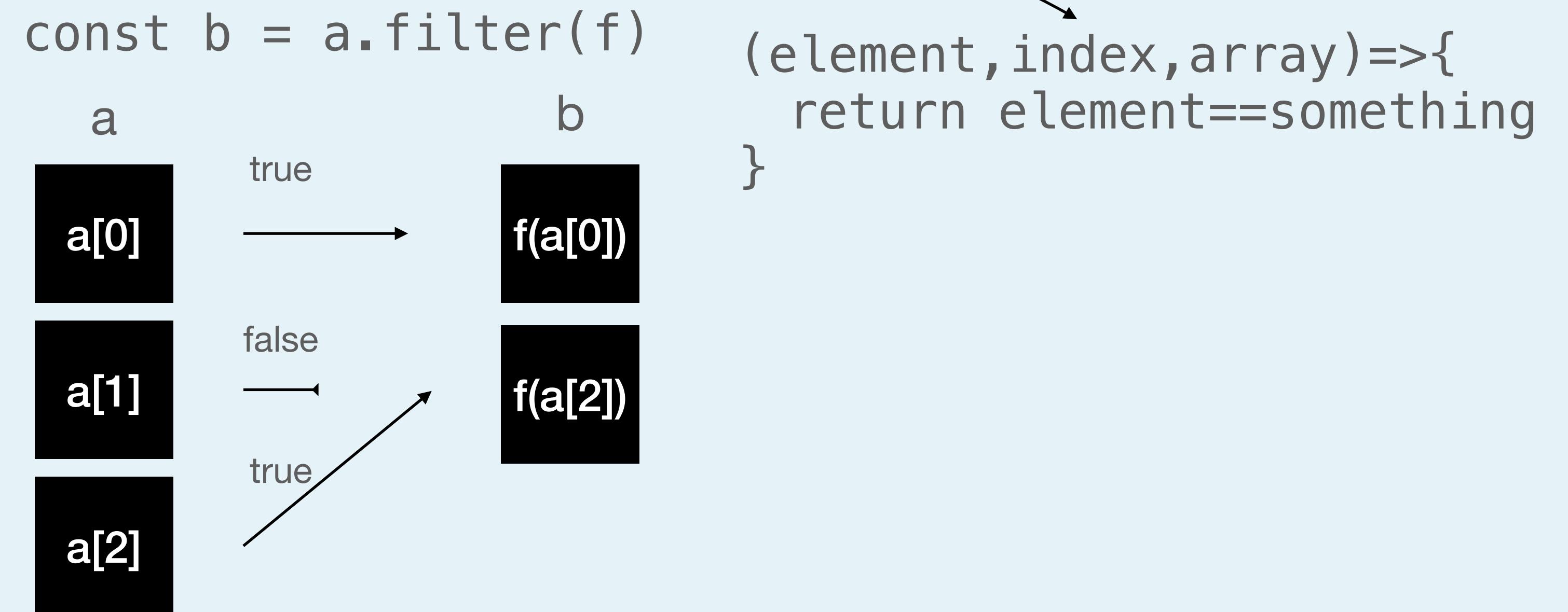
一例：全部ひらがなにしてみる

```
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

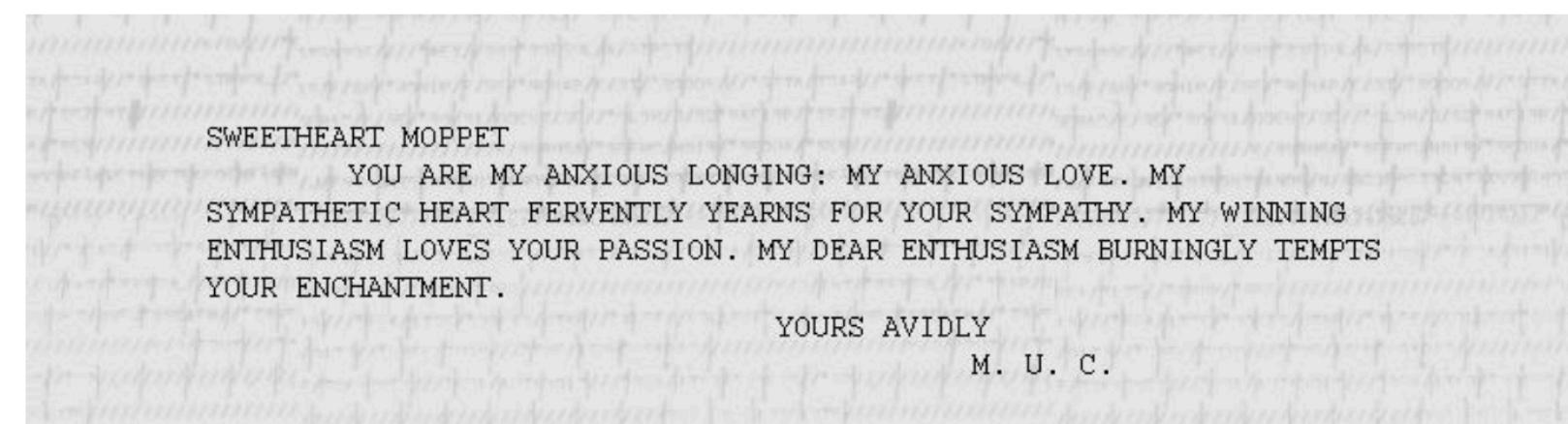
### A Queer History of Computing: Part Three

by [Jacob Gaboury](#)

2013-04-09

*For the preceding segment of this four part genealogy, see [Part 2](#)*

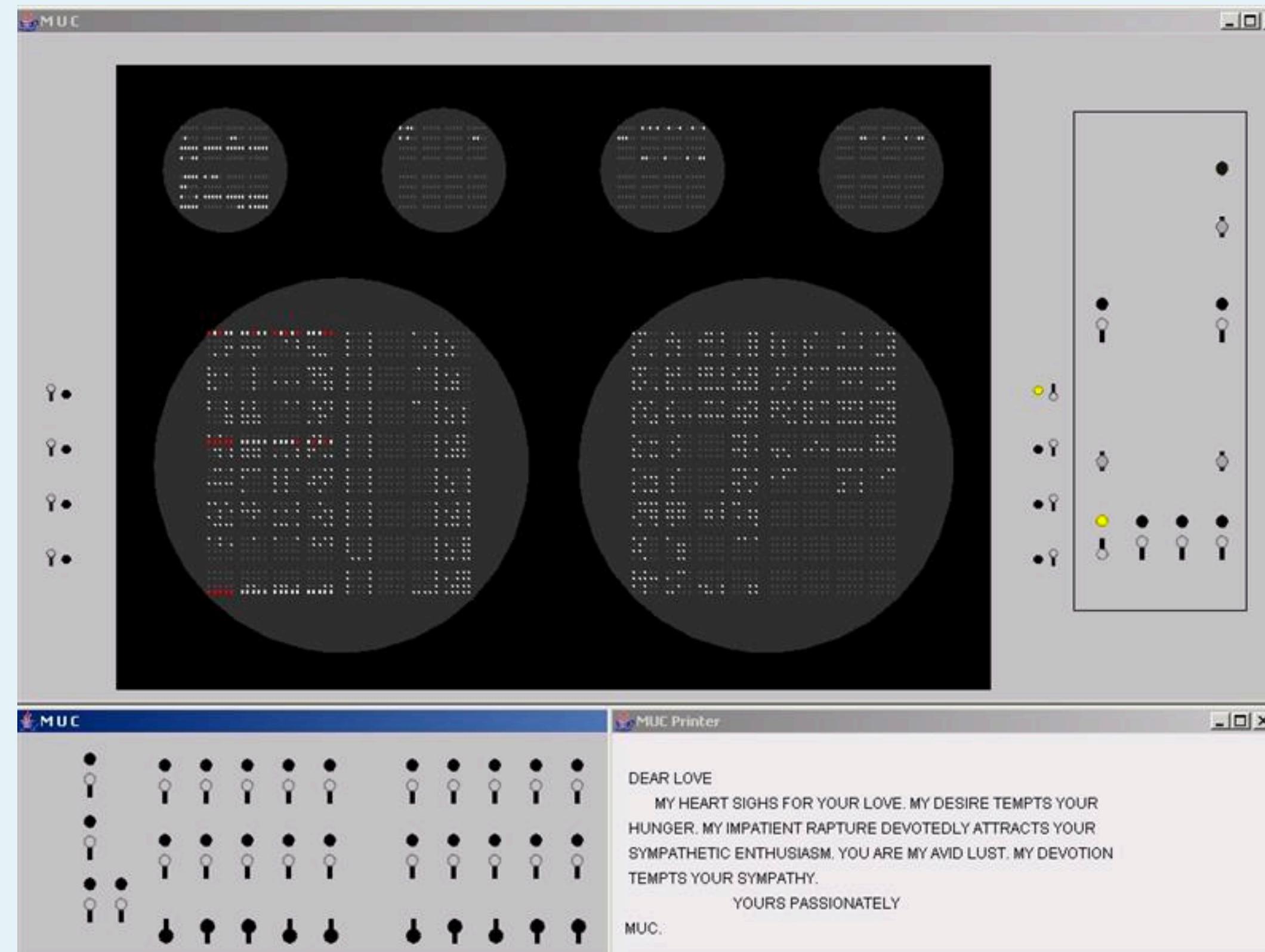
*In this third segment of [our genealogy](#) we begin to form a connection, and to examine those lesser-known but foundational figures that radiate out from Turing's early work. Perhaps appropriately, given the venue, this second figure leads us to one of the earliest examples of computational art ever 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.*



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

Matt SephtonによるPHPでの実装

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

# MUC Loveletter Generator

## Christopher Strachey, Alan Turing

The screenshot shows a web application for generating love letters. On the left, there are two main sections: "DARLING LOVE" and "DEAR DUCK". Each section contains a poem generated by the program. On the right, there is a sidebar with information about the original work and its reimplementation.

**DARLING LOVE**

MY THIRST LOVINGLY LOVES YOUR HEART. MY LONGING PRIZES YOUR AMOROUS LIKING. YOU ARE MY COVETOUS DESIRE: MY FERVENT PASSION. YOU ARE MY EROTIC SYMPATHY.

YOURS ARDENTLY

M.U.C.

DEAR DUCK

YOU ARE MY IMPATIENT LONGING. MY LONGING SEDUCTIVELY HOPES FOR YOUR CRAVING FANCY. YOU ARE MY ANXIOUS TENDERNESS. MY ANXIOUS FELLOW FEELING ADORES YOUR RAPTURE. MY AVID SYMPATHY WISTFULLY CHERISHES YOUR SWEET APPETITE.

**Love Letters**

Christopher Strachey  
1953  
Ferranti Mark 1  
Autocode

Reimplemented by  
Nick Montfort  
for Memory Slam  
Also: a [Python 2/3](#) version

The bottom part of the screenshot shows the browser's developer tools, specifically the Elements tab. It displays the HTML structure and the embedded Javascript code used to generate the love letters. The Javascript defines arrays for various words (first, second, adjectives, nouns, verbs, adverbs) and functions for random selection (rand\_range, choose).

```
<!-->
<style type="text/css">::></style>
<script type="text/javascript"> == $0
  var t = 0,
    first = ['DARLING', 'DEAR', 'HONEY', 'JEWEL'],
    second = ['DUCK', 'LOVE', 'MOPPET', 'SWEETHEART'],
    adjectives = ['ADORABLE', 'AFFECTIONATE', 'AMOROUS', 'ANXIOUS', 'ARDENT', 'AVOID', 'BREATLESS', 'BURNING', 'COVETOUS', 'CRAVING', 'CURIOUS', 'DARLING', 'LOVEABLE', 'LOVESICK', 'LOVING', 'PASSIONATE', 'PRECIOUS', 'SWEET', 'SYMPATHETIC', 'TENDER', 'UNSATISFIED', 'WISTFUL'],
    nouns = ['ADORATION', 'AFFECTION', 'AMBITION', 'APPETITE', 'ARDOUR', 'CHARM', 'DESIRE', 'DEVOTION', 'EAGERNESS', 'ENCHANTMENT', 'ENTHUSIASM', 'FANCY', 'LONGING', 'LOVE', 'LUST', 'PASSION', 'RAPTURE', 'SYMPATHY', 'TENDERNESS', 'THIRST', 'WISH', 'YEARNING'],
    adverbs = ['AFFECTIONATELY', 'ANXIOUSLY', 'ARDENTLY', 'AVIDLY', 'BEAUTIFULLY', 'BREATLESSLY', 'BURNINGLY', 'COVETOUSLY', 'CURIOUSLY', 'DEVOTEDLY', 'SEDUCTIVELY', 'TENDERLY', 'WINNINGLY', 'WISTFULLY'],
    verbs = ['ADORES', 'ATTRACTS', 'CARES FOR', 'CHERISHES', 'CLINGS TO', 'DESIRERS', 'HOLDS DEAR', 'HOPES FOR', 'HUNGRERS FOR', 'IS WEDDED TO', 'LIKES', 'TEMPTS', 'THIRSTS FOR', 'TREASURES', 'WANTS', 'WISHES', 'WOOS', 'YEARNS FOR'];

  function rand_range(maximum) {
    "use strict";
    return Math.floor(Math.random() * (maximum + 1));
  }
  function choose(array) {
    "use strict";
    return array[rand_range(array.length - 1)];
  }
</script>
```

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.html)

[https://nickm.com/memslam/love\\_letters.py](https://nickm.com/memslam/love_letters.py) Python版も

# 偶然短歌bot (いなにわ)

O 偶然短歌bot @g57577 · 3時間

友人に殴られたのをきっかけに人の目が気になるようになり #tanka  
ウィキペディア日本語版「岩井秀人」より

 ja.wikipedia.org  
岩井秀人 - Wikipedia

💬 4 ⏱ 3:26 🌐 26 📂 6,913 ⏚ 🔍

O 偶然短歌bot @g57577 · 8時間

自動

近年は日本人の体格も昔に比べ、一段と良く #tanka  
ウィキペディア日本語版「弾倉」より

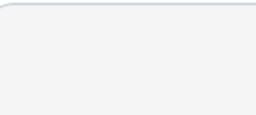
 ja.wikipedia.org  
弾倉 - Wikipedia

💬 3 ⏱ 3:26 🌐 18 📂 7,611 ⏚ 🔍

O 偶然短歌bot @g57577 · 15時間

自動

原型となった短編小説は小島の初期の作品である #tanka  
ウィキペディア日本語版「伊藤計劃」より



<https://twitter.com/g57577>

# 偶然短歌bot (いなにわ)

guuzen-tanka / pickup.rb

↑ Top

Code Blame

Raw ⌂ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉

```
95
94  def check_tanka(tanka, next_node)
95    # 括弧
96    parentheses = [0] * PARENTHESES_LEFT.length
97    tanka.each do |node|
98      idx = PARENTHESES_LEFT.index(node[:surface])
99      unless idx.nil?
100        parentheses[idx] += 1
101      end
102      idx = PARENTHESES_RIGHT.index(node[:surface])
103      unless idx.nil?
104        parentheses[idx] -= 1
105        return false if parentheses[idx] < 0
106      end
107    end
108    return false unless parentheses == [0] * PARENTHESES_LEFT.length
109    # 末尾
110    return false if "連体詞" == tanka[-1][:feature]["品詞"]
111    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)$/ =~
112      tanka[-1][:feature]["品詞細分類1"]
113    return false if "助動詞" == tanka[-1][:feature]["品詞"] and
114      "だ" == tanka[-1][:feature]["原形"]
115    return false if !next_node.nil? and
116      !check_first_word(next_node, COUNTS.length)
117    # OK
118    return true
119  end
120
121  def find_tanka(texts)
122    tankas = []
123    texts.each do |text|
124      nm_nodes = $nm.parse_as_nodes(text)
125      nodes = convert_nodes(nm_nodes)
126      nl = nodes.length
127      (0...nl).each do |i|
128        ...
129      end
130    end
131  end
132
```

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