# JXA for TypeScript/ Node.js

#### 自己紹介

• Name : azu

• Twitter:@<u>azu\_re</u>

• Website: Web scratch, JSer.info



## JXA(JavaScript for Automation)

- AppleScriptのJavaScript版(WebKit)
- macOSにはビルトインされているので何もしなくても使える
  - osascriptという実行エンジンがある
- Home · JXA-Cookbook/JXA-Cookbook Wiki
- jxa@apple-dev.groups.io | Wiki

#### JXAの問題

- クラッシュする
- リファレンスがまともにない
- 実行環境がまともではない 落ちまくる
- エディタ環境もまともになり
- デバッグが難しい
- macOSのみ対応

#### JXAのいいところ

- (JXA)スクリプトファイルでmacOSの自動化ができる
- Objective-Cと同等の表現ができる
- mac向けにネイティブアプリはなんか操作できる
- mac向けのアプリでは一部便利な関数が用意されている

#### JXAのできること

- Automatorでできること全部
- アプリへのキー入力、開く閉じるなどの操作
  - ラベルを見つけて操作など無理やりなんとかできる
- 対応アプリはアプリから情報を取得できる
  - omnifocusはTodoデータをデータベースから引っ張れるとか。
- ダイアログとかでインタラクティブなものを簡単につくれる
  - しょぼいGUIアプリみたいのが作りやすい

#### JXAの用途

- アプリ作るほどじゃないけど自動化したいときに使う
  - シェルスクリプトのアプリ向け処理みたいな感じで使う
- macOSネイティブなので、Alfredなど対応アプリも多い
  - osascript コマンドで実行できるのでシェル叩けるならどこからで も
  - メニューバーからも実行できるようにもなってる
  - AppleScriptをメニューバーから実行するための設定

#### スクリプト言語としてのJXA

- JavaScriptエンジンはmacOsに依存(WebKit/JavaScriptCore)
- 癖のあるJavaScriptフレームワーク上のJavaScriptという印象
  - プロパティがgetterじゃなくてname()のような関数呼び出し
- AppleScriptより扱いやすい
  - 非英語圏の人間にとってはAppleScriptの構文が難しすぎる
  - Objective-Cよりはるかに英語的な構文

#### AppleScriptでChromeでGoogleを開く

```
set myLink to "https://google.com/"
tell application "Google Chrome"
    tell its window 1
        set theTabs to count of tabs
        set URL of tab 1 to myLink
    end tell
end tell
```

#### AppleScriptは英語

次の3行はAppleScriptでは同じ意味(itはtellで指定したオブジェクトなどを表す)

```
handlerX() of it
it's handlerX()
its handlerX()
```

次の2行はAppleScriptでは同じ意味(count関数に直接引数"abc"を渡して呼ぶ。 前後どちらも対応)

```
count "abc"
<u>"ab</u>c" count
```

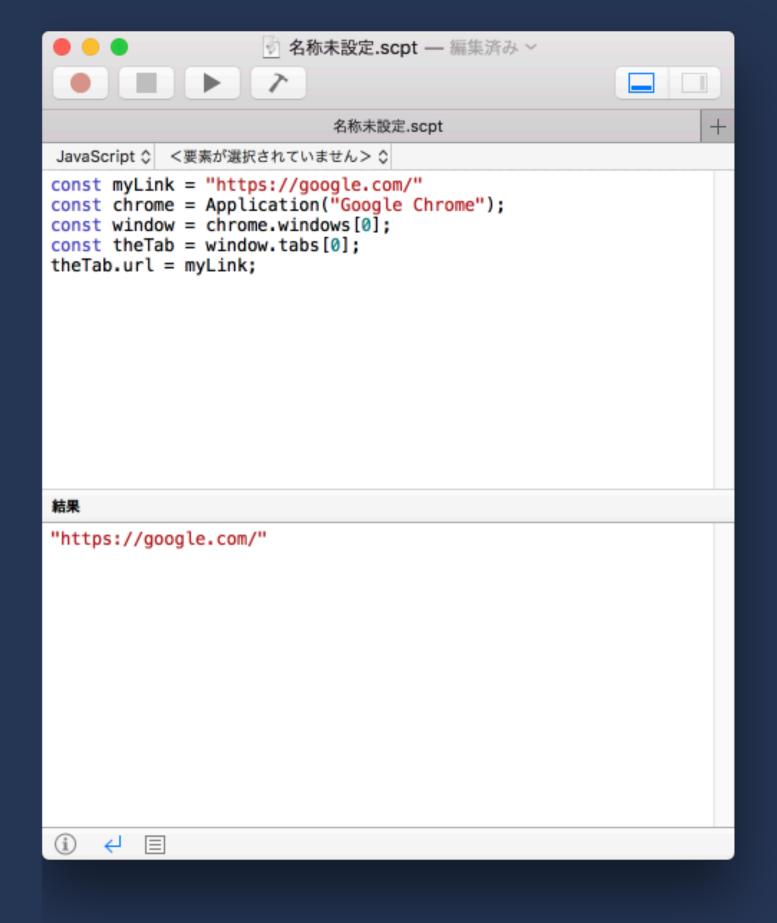
鳶嶋工房 / AppleScript / Tips / 用語の同義語 より

#### JXAでChromeでGoogleを開く

```
const myLink = "https://google.com/"
const chrome = Application("Google Chrome");
const window = chrome.windows[0];
const theTab = window.tabs[0];
theTab.url = myLink;
```

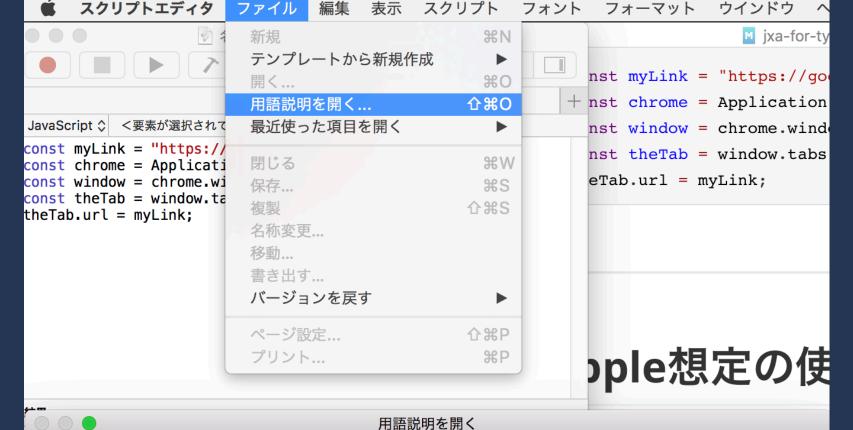
#### Apple想定の使い方

- スクリプトエディタ
  - エディタかつ実行環境



### Apple想定の使い方 - リファレンス

- スクリプトエディタから"用語説明"でインストールしてるア プリのリファレンスが見られる
- 鳶嶋工房 / AppleScript / 入門 / 用語説明の読み方(命令編)
- 鳶嶋工房 / AppleScript / 入門 / 用語説明の読み方(値編)



#### 用語説明を開く項目を選択:

	名前	^	種類	バージョン	パス
	グラブ.app		アプリケーション	1.9	/Applications/Utilities/Grab.app
WARNES AY 7136	コンソール.app		アプリケーション	1.0	/Applications/Utilities/Console.app
	システム環境設定.app		アプリケーション	14.0	/Applications/System Preferences.app
<b>\$</b>	システム情報.app		アプリケーション	10.12	/Applications/Utilities/System Information
	スクリプトエディタ.app		アプリケーション	2.9	/Applications/Utilities/Script Editor.app
>-	ターミナル.app		アプリケーション	2.7.4	/Applications/Utilities/Terminal.app
	テキストエディット.app		アプリケーション	1.12	/Applications/TextEdit.app
B!	はてなブックマーク検索PWA.app		アプリケーション	2018.4.1	/Users/azu/Library/Application Support/C
B!	はてなブックマーク検索PWA.app		アプリケーション	2018.5.2	/Users/azu/Library/Application Support/C
4	フォルダアクション設定.app		アプリケーション	1.2	/System/Library/CoreServices/Applicatio
=	プレビュー.app		アプリケーション	9.0	/Applications/Preview.app
3	マップ.app		アプリケーション	2.0	/Applications/Maps.app
	メール.app		アプリケーション	10.3	/Applications/Mail.app
<u></u>	メッセージ.app		アプリケーション	10.0	/Applications/Messages.app
	メモ.app		アプリケーション	4.4	/Applications/Notes.app
	リマインダー.app		アプリケーション	4.0	/Applications/Reminders.app
	画面共有.app		アプリケーション	1.7.1	/System/Library/CoreServices/Applicatio
*	写真.app		アプリケーション	2.0	/Applications/Photos.app
	連絡先.app		アプリケーション	10.0	/Applications/Contacts.app

ブラウズ... キャンセル

セル

選択

selectAll P loading cutSelection copySelection pasteSelection undo redo stop viewSource execute Application C Tab BookmarkFolder BookmarkItem **Tab** Object : A tab. ELEMENTS contained by windows. PROPERTIES id (integer, r/o): Unique ID of the tab. title (text, r/o): The title of the tab. url (text): The url visible to the user. loading (boolean, r/o) : Is loading? undo, redo, cutSelection, copySelection, pasteSelection, selectAll, goBack, goForward, reload, stop, print, viewSource, save, close, execute. BookmarkFolder Object: A bookmarks folder that contains other bookmarks folder and bookmark items. contains bookmarkFolders, bookmarkItems; contained by application, bookmarkFolders. id (number, r/o): Unique ID of the bookmark folder. title (text): The title of the folder. index (number, r/o): Returns the index with respect to its parent bookmark folder BookmarkItem Object: An item consists of an URL and the title of a bookmark contained by bookmarkFolders. id (integer, r/o): Unique ID of the bookmark item. title (text): The title of the bookmark item. url (text): The URL of the bookmark. index (number, r/o): Returns the index with respect to its parent bookmark folder

Google Chrome.sdef

JavaScript

reload

goBack

goForward

プリント

P id

title

P url

Q用語

検索

戻る/進む テキストサイズ

Standard Suite

Chromium Suite

#### JXAのデバッグ

- Safariを開く
- JXAでdebugger;をいれて実行する
- Safariでデバッグできる
- コンソールが使えるのでREPLライク

#### debugger;

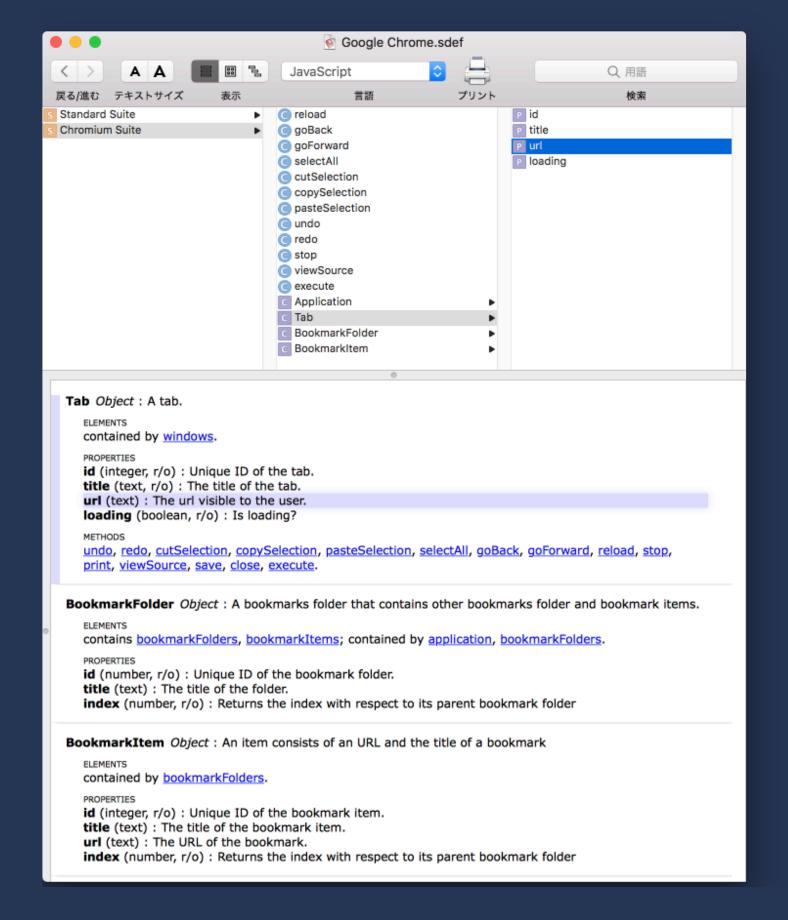
JavaScriptによるMacの自動操作入門の情報まとめ | 大石制作ブログ

#### JXAの問題

- リファレンスが読めない
  - リファレンスに載ってないものが多すぎる
- まともなエディタ環境がない
  - コードの補完が効く環境がないに等しい
- デバッグが難しすぎる
  - 落ちる、クラッシュ、リファレンスない
- Node.jsでできるところはNode.jsでやりたい
  - pathとか文字列処理とか、JXAはライブラリの仕組みはあるけどnpmは使えない

#### リファレンス

- リファレンス自体の読み方が難しい
  - Tabというクラスは定義されているが、Tab を得る方法が載ってない...
  - すべてのプロパティやコマンドなどが載っているわけじゃない(自動生成されているレベル)
- リファレンスは.sdefという複雑なXMLで書かれている
  - クラスの定義、enumの定義、定数、 interfaceなどやたら高度な定義ができる XML



#### リファレンスから型定義を生成する

- TypeScriptの型定義ファイルがあれば今どきのエディタは補 完が効く
- リファレンス(.sdef)はある種の型定義XMLに見えた
- パースしてXMLから.d.tsを作ればいい
- 作った: => @jxa/sdef-to-dts

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE dictionary SYSTEM "file://localhost/System/Library/DTDs/sdef.dtd">
                                                                 export namespace StandardAdditions {
                                                                   // Default Application
<dictionary>

→|<suite_name="User_Interaction".code="syso".description="Basic.commands.</p>
                                                           4
                                                                 export interface Application {}
 for interacting with the user">
                                                                  ...//. Class
>><command_name="beep".code="sysobeep".description="Beep_1.or.more.</p>
                                                                 /**
 times">
                                                                  * This file was automatically generated by json-schema-to-typescript.
* DO NOT MODIFY IT BY HAND. Instead, modify the source JSONSchema file,
 description="number of times to beep"/>
                                                                  * and run json-schema-to-typescript to regenerate this file.
→ </command>
                                                           10
an application on this machine or the network">
                                                           12
                                                                    object specified with a POSIX (slash)-style pathname.
optional="yes" description="the dialog window title"/>
export interface POSIXFile {
 optional="yes" description="the prompt to be displayed in the dialog box"/>
17
                                                                    * the POSIX (slash)-style path of a file or alias object
 type="boolean" optional="yes" description="Allow multiple items to be
                                                           18
 selected? (default is false)"/>
                                                           19
                                                                   POSIXPath(): string;
20
 description="the desired type of result. May be application (the default)
                                                           21
 or alias."/>
                                                           22
23
                                                                  * This file was automatically generated by json-schema-to-typescript.
                 <xref target="choose remote application"/>
                                                           24
                                                                  * DO NOT MODIFY IT BY HAND. Instead, modify the source JSONSchema file,
→ </command>
                                                                  * and run json-schema-to-typescript to regenerate this file.
                                                           25
26
 color">
                                                           27
28
                                                                 /**
 optional="yes" description="the default color"/>
                                                           29
                                                                  * A. Uniform Resource Locator or Uniform Resource ID (URI)
30
→ </command>
                                                                 export interface URL {
                                                           31
M<command_name="choose file" code="sysostdf" description="Choose a</p>
                                                           32
 file on a disk or server">
                                                           33
                                                                    * property that allows getting and setting of multiple properties
 34
 optional="yes" description="the prompt to be displayed in the dialog box"/>
                                                           35
                                                                   properties(): any;
36
 description="a list of file types or type identifiers. Only files of the
                                                           37
                                                                    * a name given to this URL, usually the name of the page it refers to
 specified types will be selectable.">
                                                           38
>> > > > type="text".list="yes"/>
                                                                   name(): string;
                                                           39
```

#### リファレンス to 型定義のテスト

- 型定義(TypeScript)への変換が上手くいってるのかをテストする必要がある
  - でもこれがJXA触ったの初めて全くわからない
- 既存のJXAで書かれたコードに型を適応すればOK
- Mac: JavaScript for Automation (JXA) 例文辞典のサンプルコードを全部ダウンロードして、型チェックして通すテストを書きながら開発した

# JXAの型定義ファイルを使ってコード 補完する

- macOSに入ってるアプリなどのリファレンスからTypeScriptの型定義を生成
  - @jxa/types \( \sum\_{\text{@jxa/global-type}} \)
- @jxa/qlobal-typeはg1oba1にJXAのグローバルオブジェクトの定義を追加するモジュール
- 読み込めば、ざっくりとした補完 + 型チェックが効くようになる
  - リファレンス自体に載ってないものが多いためanyで逃げている...

```
import "@jxa/global-type";
// your JXA application
var userName = Application("System Events").currentUser().name();
```

```
example.ts ×
       import "@jxa/global-type";
  10
  12
  13
  14
```

#### JXAのデバッグ

- console.logするとスクリプトエディタが落ちる
- SafariにつないでもSafariごと落ちる
- CLIのREPLから叩くと落ちても安全
- 公式のREPLがosascript にビルトインされている

#### REPLを起動する

\$ osascript -il JavaScript

#### JXAの自作REPL

- osascript にREPLがあるのをしらなくてNode.jsで動くREPLを 書いた
- const repl = require('repl');
  - REPL | Node.js v10.5.0 Documentation
- 作ったREPL: @jxa/repl
- \$ npx @jxa/repl
- > Application("System Events").currentUser().name();
  'azu'

#### JXAのデバッグ

- JXAはとにかくデバッグしにくい
- JXAで書かなければ解決する
- JXAがいらない部分はNode.jsで書こう
- => Node.jsの中でJXAを実行すればいい

### JXA in Node.js

- @jxa/run
  - Node.jsからJXAを実行して結果を取得するモジュール
- 先駆者
  - sindresorhus/run-jxa: Run JXA code and get the result
  - wtfaremyinitials/osa2: Interact with Apple's Open
     Scripting Architecture in node.js

#### @jxa/run

```
import "@jxa/global-type";
import { run } from "@jxa/run";
export const currentUserName = () => {
   // このコールバック関数内はJXAとしてosascriptで実行される
   return run(() => {
       const sys = Application("System Events");
       return sys.currentUser().name();
   });
// メインとかは普通にNode.js
export const example = async () => {
   const userName = await currentUserName();
   return `User: ${userName}`;
};
```

### JXA in Node.js

- JXAが最小限になる
  - 最小限のJXAは書かないといけないのは同じだけど...
- メインがNode.jsになってデバッグしやすい
- npmを簡単に組み合わせることができてエコシステムが拡張できる
- JXAで書いたものをモジュール化できる(npmで配れる)
  - つかえるのはmacOSだけなのは同じだけど

```
export type ModifierOption = {
   shift?: boolean;
   control?: boolean;
   option?: boolean;
   command?: boolean;
};
function createModifier(modifierOption: ModifierOption) {
   const modifiers = [];
    if (modifierOption.shift)
       modifiers.push("shift down");
    if (modifierOption.command)
       modifiers.push("command down");
    if (modifierOption.control)
       modifiers.push("control down");
    if (modifierOption.option)
       modifiers.push("option down");
   return modifiers;
// キーストロークを送信する関数 from JXA in Node.js
export function sendKeyStroke(key: string, modifierOption: ModifierOption) {
   const modifiers = createModifier(modifierOption);
   return run((key, modifiers) => {
       const SystemEvents = Application("System Events");
       SystemEvents.keystroke(key, { using: modifiers });
    }, key, modifiers);
```

#### ユースケース

- Firefoxで開発者ツールを開いた状態でのスクリーンショット を撮って保存する
  - Seleniumだと開発者ツールを開けない?
  - screenshot-dev-tools.ts

#### まとめ

- JXA in Node.jsをやるためにいろいろ書いた
- JXA-userland/JXAのmonorepoで公開
- Integration JXA with <u>TypeScript</u>
- Run JXA from Node.js
  - See @jxa/run and @jxa/repl
- Support Auto complete for editor/IDE via TypeScript definition file(.d.ts)
  - See @jxa/types and @jxa/global-type

#### 似た概念

- NILScript
  - WindowsでSpiderMonkeyを使ったもの
  - とても良くできていて便利だった