再考ですかーっ！？

JavaScript

SpiderMonkeyといっしょ

module.jp

概要

- 言語としてのJavaScriptをおさらい
  - 基本的なSyntax
  - Object Oriented Language JavaScript
- JavaScriptの処理系を愉しむ
  - Mozillaの処理系”SpiderMonkey”概要
  - JavaScriptを俺色に染める

module.jp

JavaScriptとは？

module.jp

JavaScriptとは？

- Netscape社がWebブラウザおよびサーバ用に開発したオブジェクト指向スクリプト言語

module.jp

JavaScriptとは？

- Netscape社がWebブラウザおよびサーバ用に開発したオブジェクト指向スクリプト言語
- MicrosoftがマネしてJScriptとか
  - 結果Webブラウザによって実装ばらばら

module.jp

JavaScriptとは？

- Netscape社がWebブラウザおよびサーバ用に開発したオブジェクト指向スクリプト言語
- MicrosoftがマネしてJScriptとか
  - 結果Webブラウザによって実装ばらばら
- ECMAがECMAScript(ECMA-262 Edition3)として標準化し、ISO/IEC 16262に承認
  - 各社ブラウザ、ソフトウェアが準拠
  - 相変わらずDOMまわりは混沌としてるけど

module.jp

C言語系Java風味の構文

- var message = “Hello World!”;
- if, else, else if, for, while, do while, switch
  - continue, break, goto
- function name(arg1, arg2) { ... },
  function (args) { ...}
- try, throw, catch, finally
- eval(...)
- +, -, *, %, ++, --...
- =, +=, -=, *=, %=...
- ==, <, >, <=, >=, !=...

module.jp

Built-in Objects

- Function
- Array
- String
- Boolean
- Number
- Math
- Date
- RegExp

```
var msg = new String(”Hello World!”);
var msg = “Hello” + “ World!”;

var list = new Array(”Perl”, “Java”);
list.length;
list[1] += “Script”;

var today = new Date();
today.toString();
```

module.jp

RegExp Object

- わりとモダンな正規表現
- String Objectにも関連メソッド多数

var reg = new RegExp(”pattern”, “gi”);
var reg = /pattern/gi;

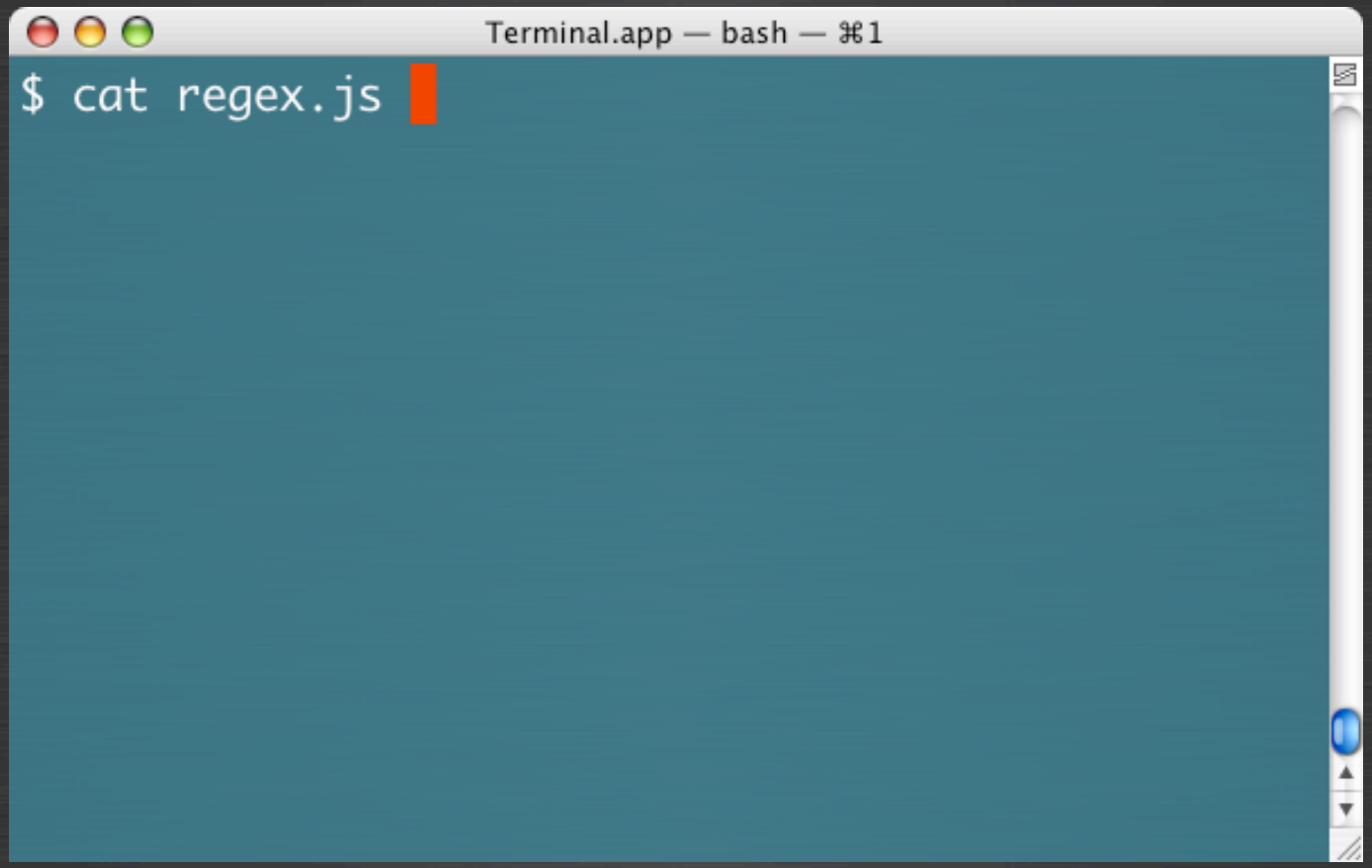
¥b, ¥B, ¥s, ¥S
¥d, ¥D, ¥w, ¥W, .
[...], [^...]
*, +, ?, {n}, {n,}, {n, m}
^, $

module.jp

RegExp Object

module.jp

RegExp Object



module.jp

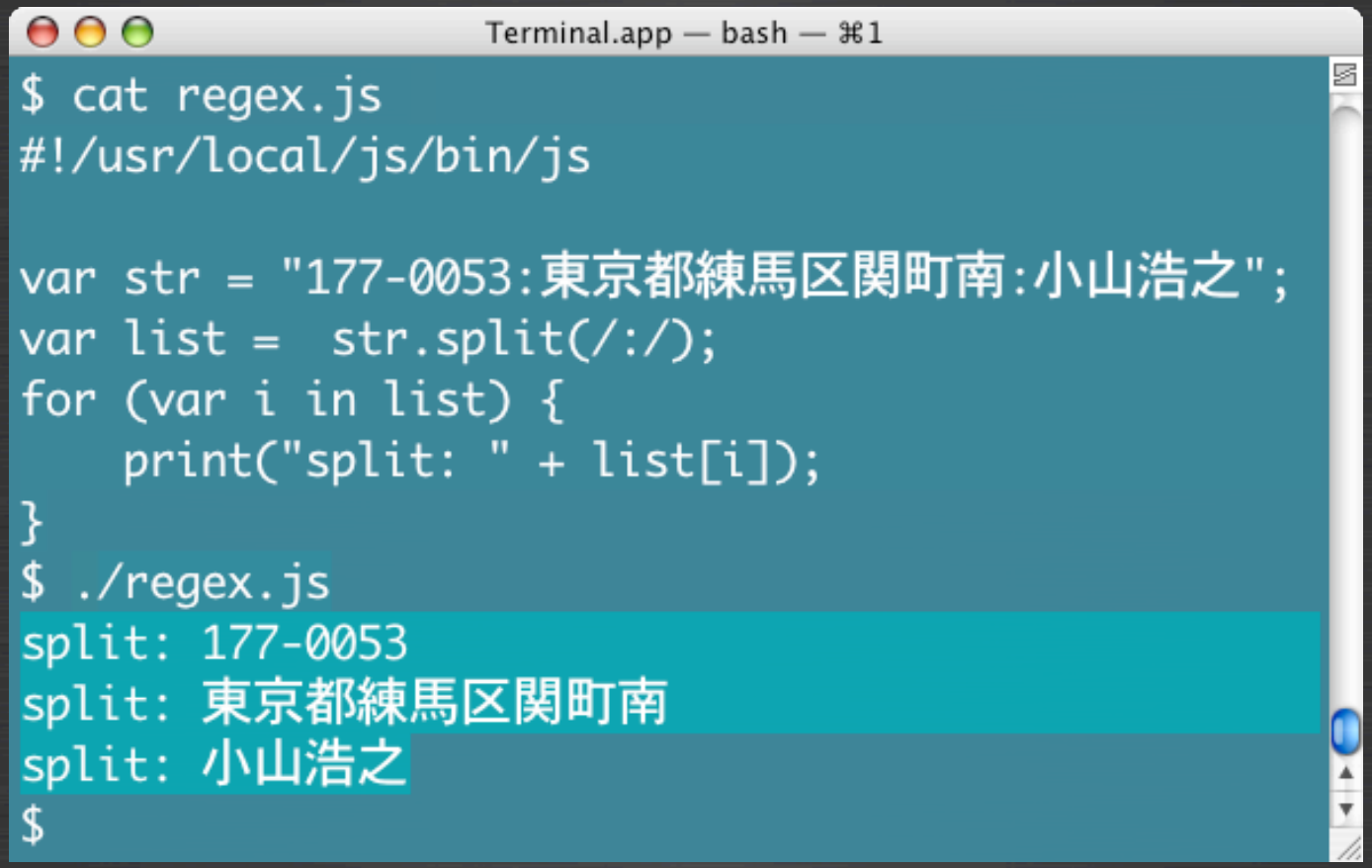
RegExp Object

module.jp

RegExp Object

module.jp

RegExp Object

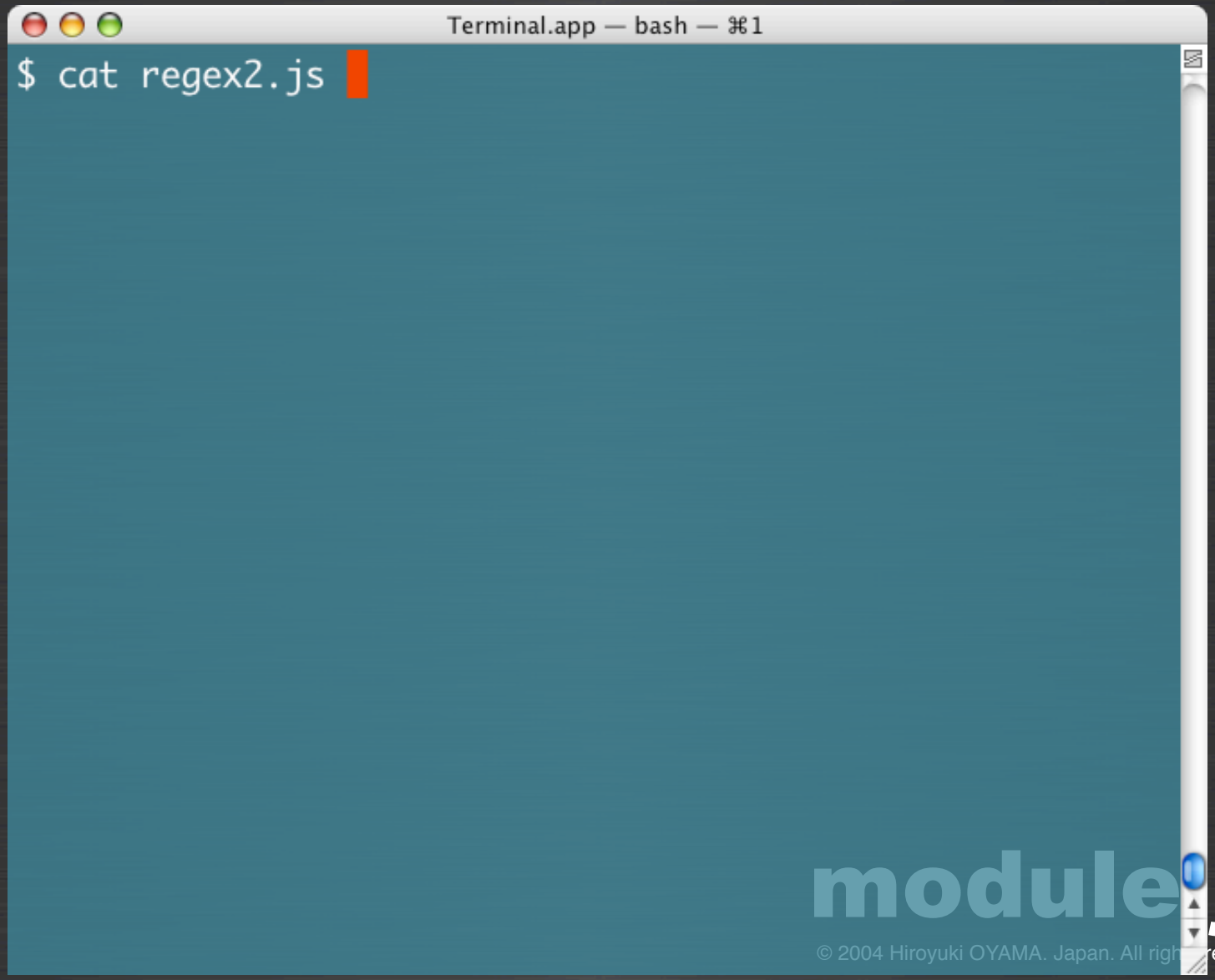


module.jp

RegExp Object

module.jp

RegExp Object



RegExp Object

Terminal.app — bash — ⌘1

```
$ cat regex2.js
#!/usr/local/js/bin/js

var reg = new RegExp("([\x21-\x7E]+|"
                     + "(\xA4[\xA1-\xF3])+|"
                     + "(\xA5[\xA1-\xF6])+|"
                     + "([\xB0-\xF4][\x00-\xFF])+)", "g");
var str = "漢字カナASCIIまじりの文字列";
var words = str.match(reg);
for (var i in words) {
    print(words[i]);
}
$
```

RegExp Object

Terminal.app — bash — ⌘1

```
$ cat regex2.js
#!/usr/local/js/bin/js

var reg = new RegExp("([\x21-\x7E]+|"
                     + "(\xA4[\xA1-\xF3])+|"
                     + "(\xA5[\xA1-\xF6])+|"
                     + "([\xB0-\xF4][\x00-\xFF])+)", "g");
var str = "漢字カナASCIIまじりの文字列";
var words = str.match(reg);
for (var i in words) {
    print(words[i]);
}
$ ./regex2.js
```

RegExp Object

Terminal.app — bash — ⌘1

```
$ cat regex2.js
#!/usr/local/js/bin/js

var reg = new RegExp("([\x21-\x7E]+|"
                     + "(\xA4[\xA1-\xF3])+|"
                     + "(\xA5[\xA1-\xF6])+|"
                     + "([\xB0-\xF4][\x00-\xFF])+)", "g");
var str = "漢字カナASCIIまじりの文字列";
var words = str.match(reg);
for (var i in words) {
    print(words[i]);
}
$ ./regex2.js
漢字
カナ
ASCII
まじりの
文字列
$
```

Object Oriented JavaScript

- プロトタイプベースのオブジェクト指向言語
- 継承ではなくて委譲

```
SubClass.prototype = new SuperClass();
function SubClass () {
    this.myMethod = function () {
        print("Hello World!");
    }
}
```

module.jp

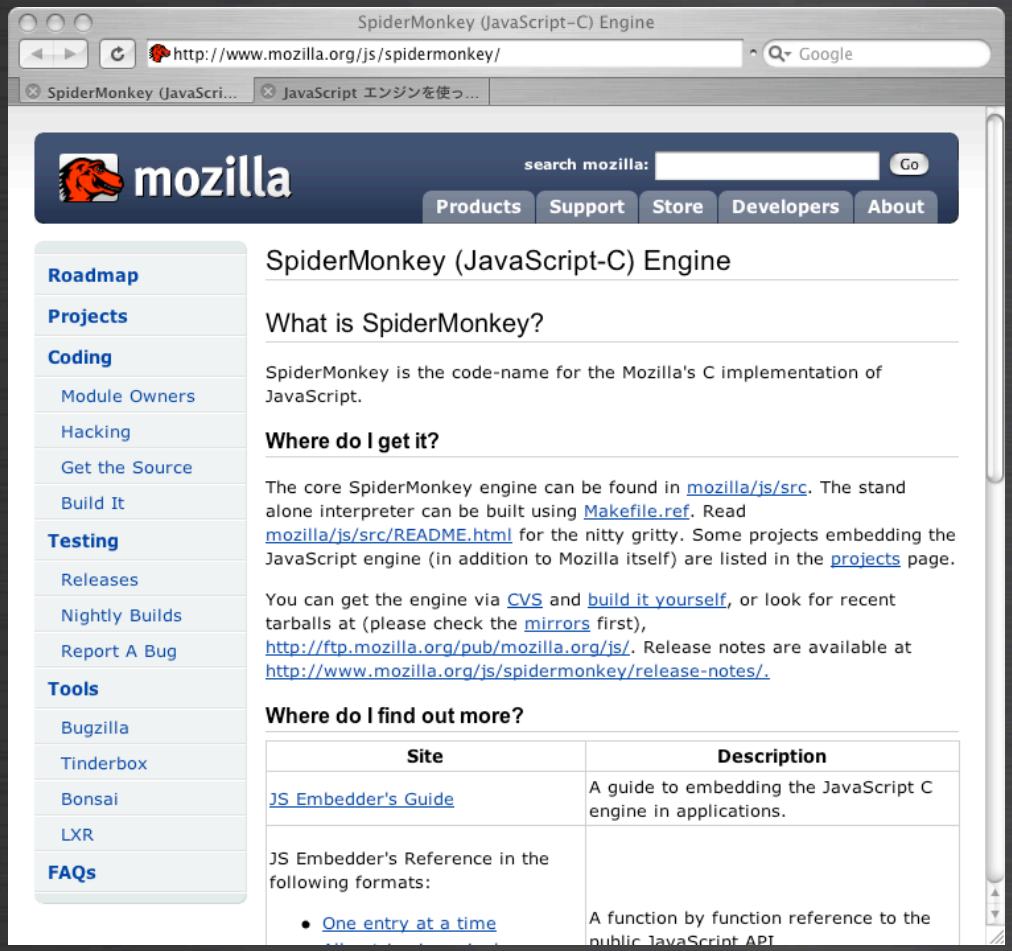
SpiderMonkey

- SpiderMonkey (JavaScript-C) Engine
- MozillaのJavaScript実装
  - 単体のパッケージとして利用可能
    - Netscape Enterprise Server
    - K-3D (3Dモデリング・アニメーション)
  - スタンドアローンのインタプリタが付属
    - Fileオブジェクト(独自拡張Class)
    - JavaとPerlのBinding付属

module.jp

SpiderMonkey配布元

http://www.mozilla.org/js/spidermonkey/



module.jp

ビルド手順

- Mozilla用のビルドシステムなので使いにくい
  - プラットフォーム決め撃ち条件コンパイル
  - 自分でLibtoolを作らない
- 勝手にGNU Autotools化してしまえ！
  - configure.in
  - Makefile.am
  - aclocal & libtoolize & automake & autoheader& autoconf

module.jp

ビルド手順

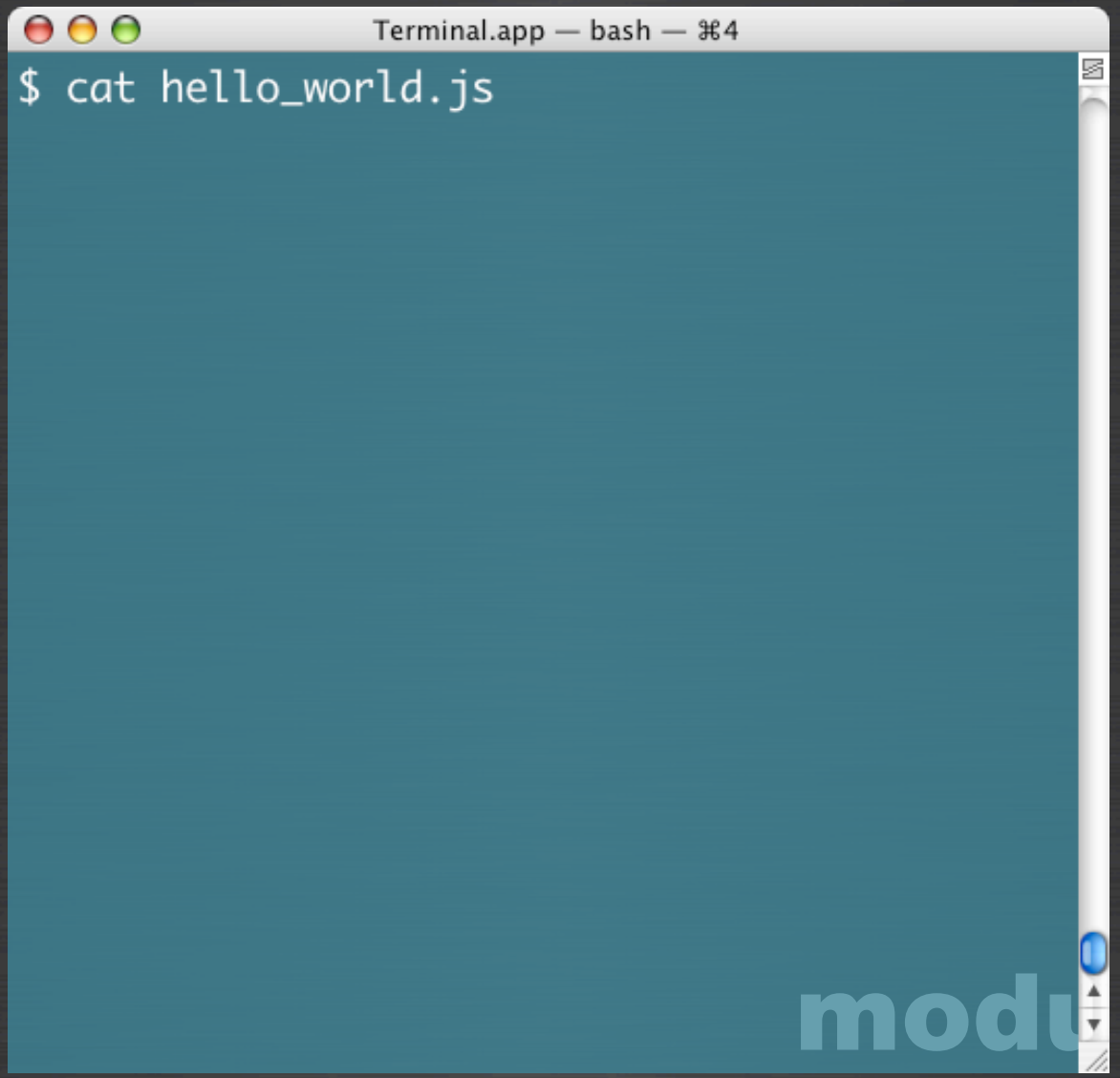
- http://module.jp/blog/autotoolize_spidermonkey.html
- configureでプラットフォームを判定して、#defineしてやる
  - AC_DEFINE(XP_UNIX)
  - AC_DEFINE(SYS4)
  - AC_DEFINE(SYSV)
  - AC_DEFINE(_BSD_SOURCE)
  - AC_DEFINE(POSIX_SOURCE)
  - AC_DEFINE(DARWIN)

module.jp

スタンドアローンのインタプリタ

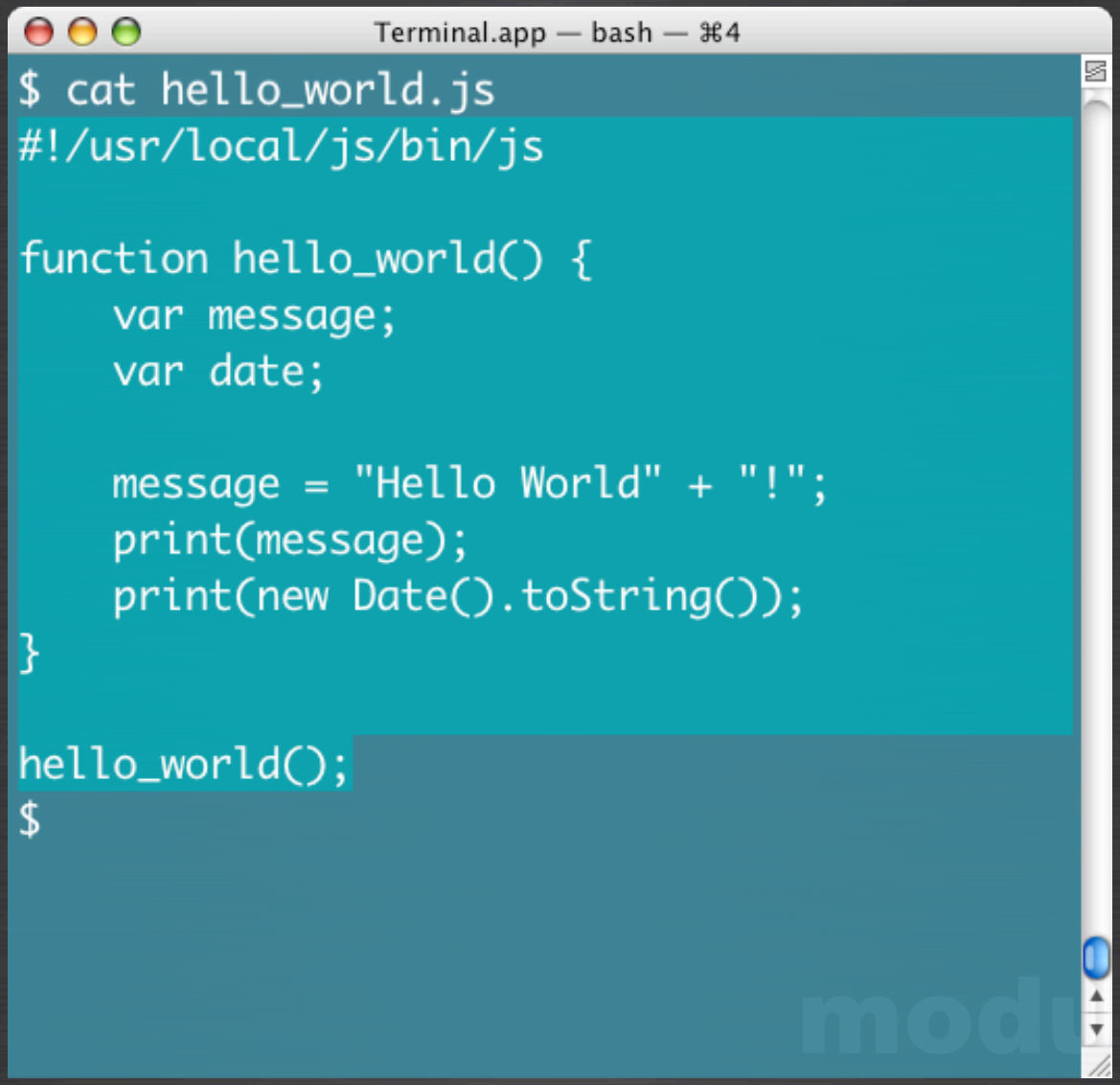
module.jp

スタンドアローンのインタプリタ



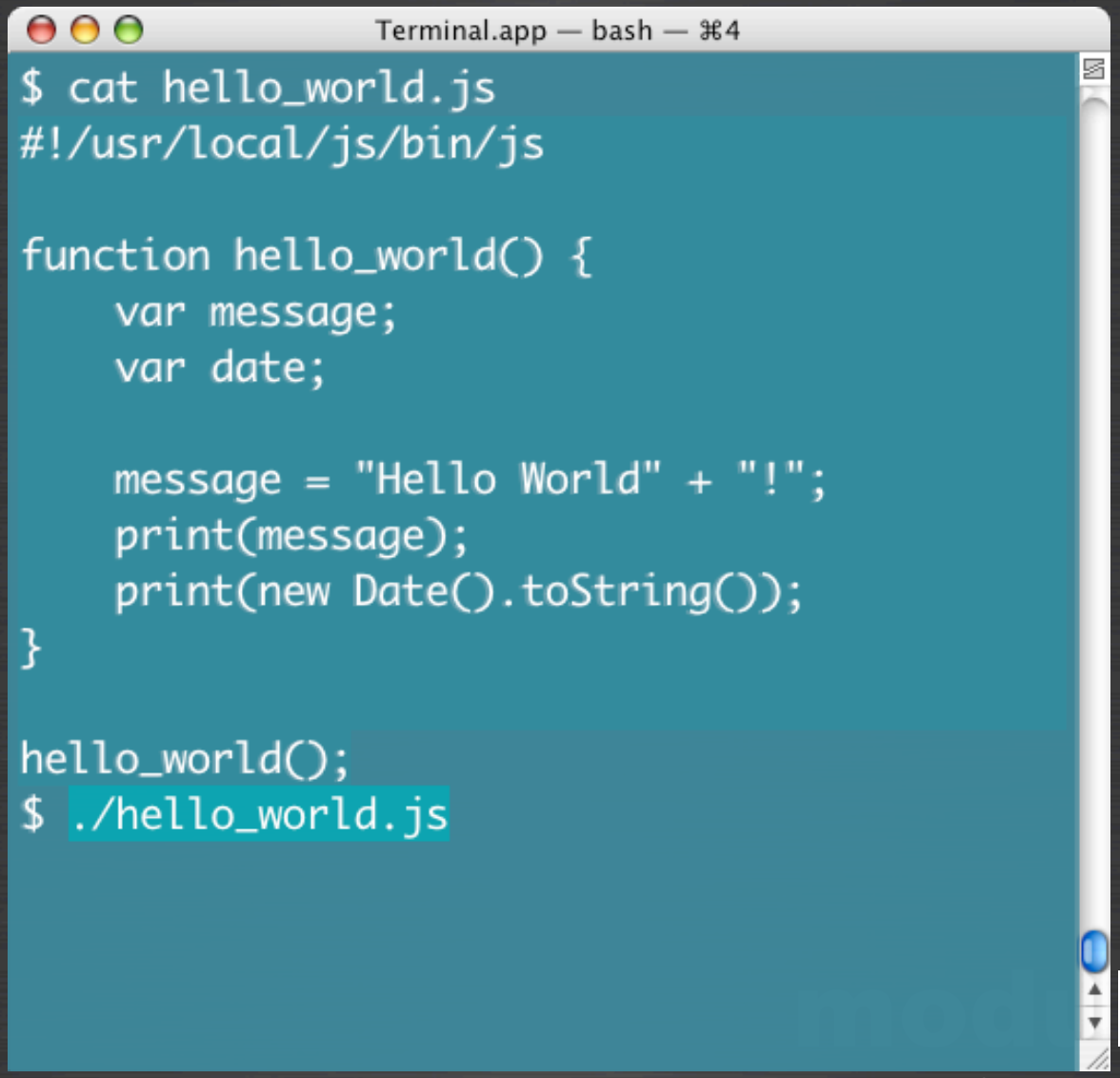
module.jp

スタンドアローンのインタプリタ



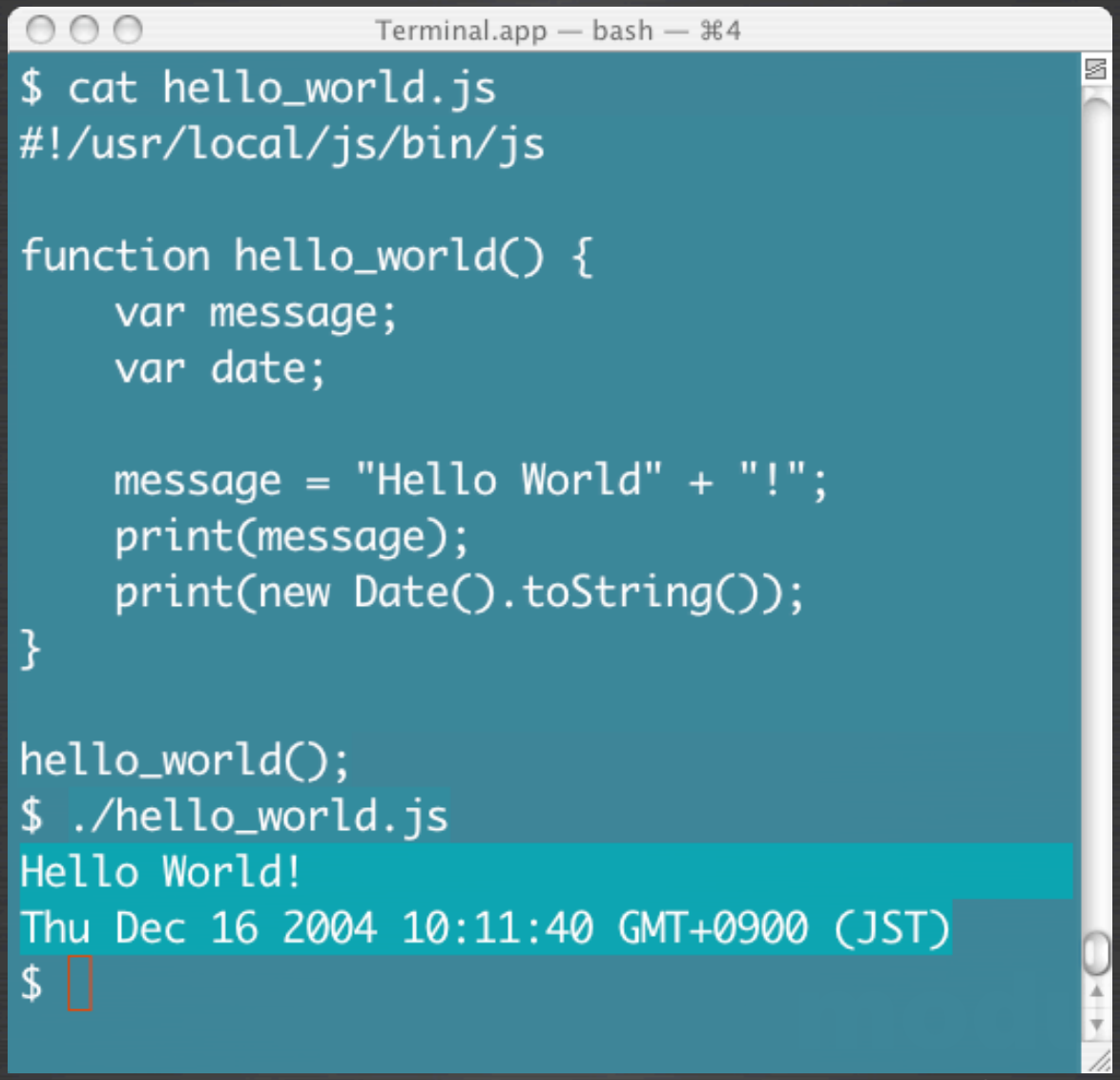
module.jp

スタンドアローンのインタプリタ



le.jp

スタンドアローンのインタプリタ



le.jp

とりあえずCから使うには？

- **jsapi.h**をincludeせよ！
- ランタイムとコンテキストと標準クラスの初期化
  - ***JSRuntime*** *rt = **JS_NewRuntime**(*RUNTIME_SIZE*);
  - ***JSContext*** *ctx = **JS_NewContext**(*rt, STACK_SIZE*);
  - ***JSObject*** *global = **JS_NewObject**(*ctx, &GLOBAL_CLASS, NULL, NULL*);
  - **JS_InitStandardClasses**(*ctx, global*);
- **libjs**をリンクせよ！

module.jp

スクリプトを実行するには？

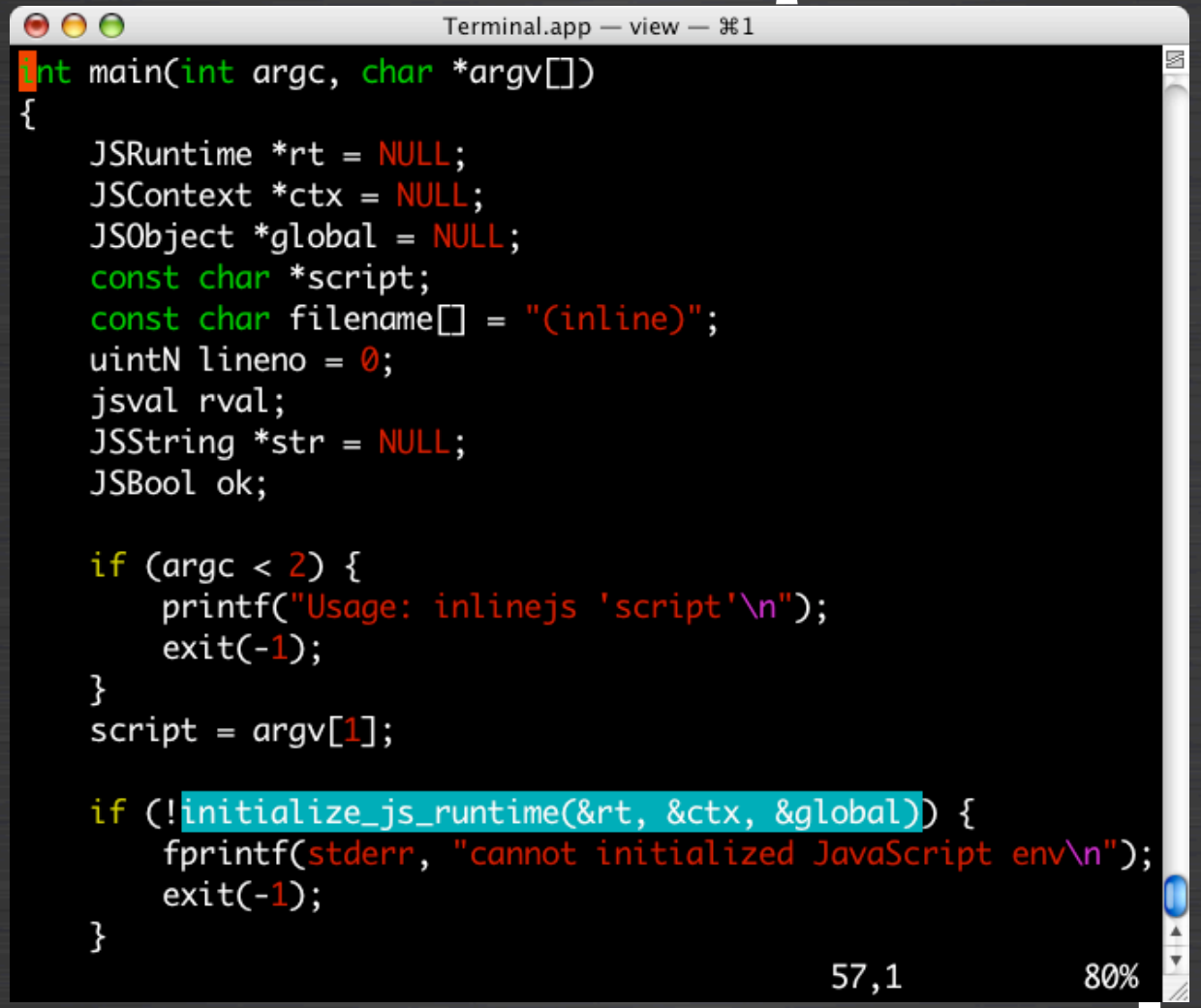
- JSContextとJSObject(のglobal object)を用意して、evalする。戻り値は***jsval*** rvalで取得する。
  - **JS_EvaluateScript**(*ctx, global, SCRIPT_STR, strlen(SCRIPT_STR), filename, linenum, &rval*);
  - ***JSString*** *str = **JS_ValueToString**(*ctx, rval*);
  - printf("result: '%s'", **JS_GetStringBytes**(*str*));
- **JS_DestroyContext**(*ctx*);　/* あとしまつ1 */
- **JS_DestroyRuntime**(*rt*);　/* あとしまつ2 */

module.jp

CからJavaScriptを呼び出す

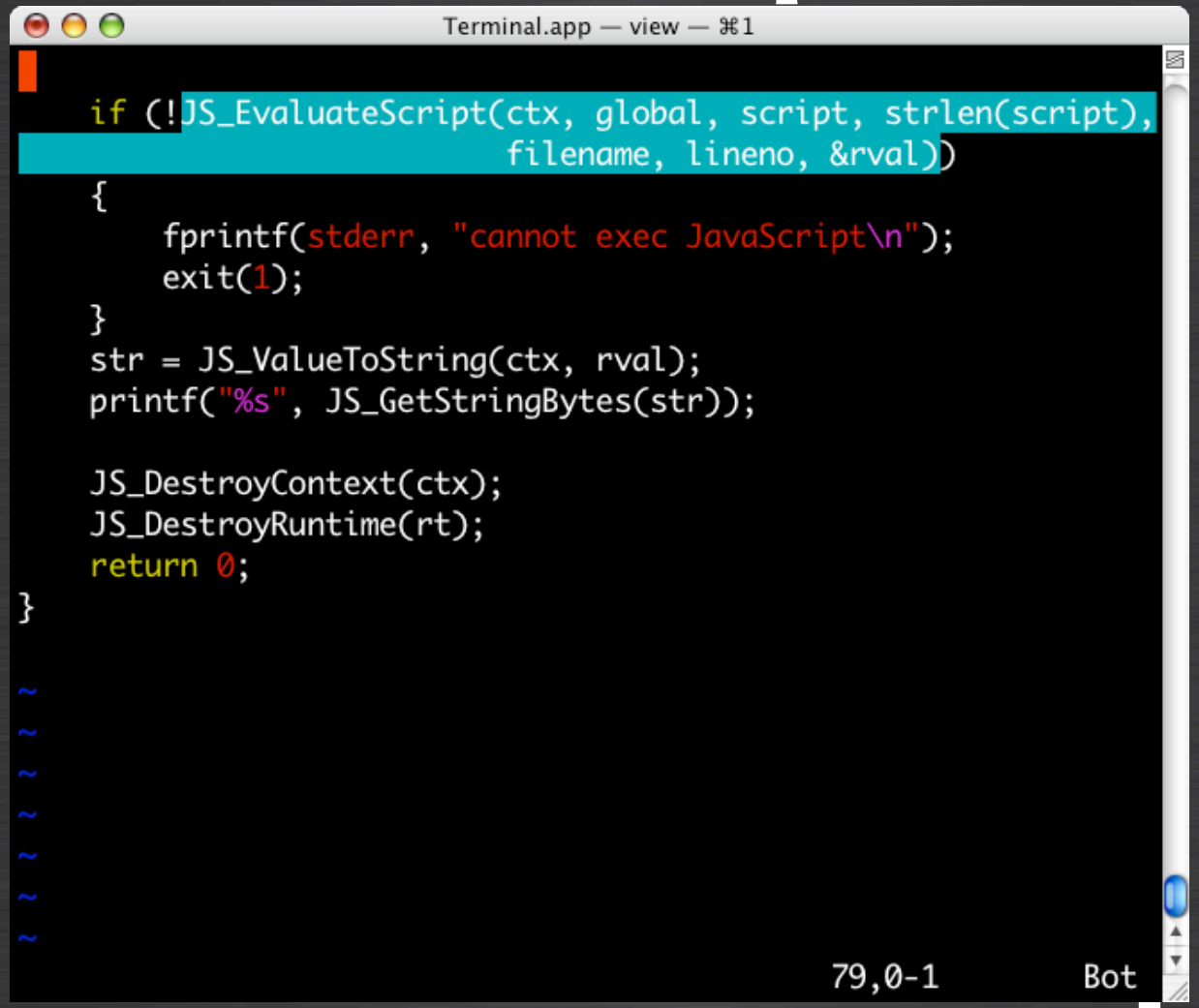
module.jp

CからJavaScriptを呼び出す



module.jp

CからJavaScriptを呼び出す



module.jp

独自関数の追加

- インタプリタにCの関数を登録して JavaScriptから実行できる。
  - JSBool ***myfunc***(JSContext *ctx, JSObject *obj, uintN argc, **jsval *argv, jsval *rval**);
  - JSFunctionSpec構造体(の配列)を定義
    - 関数名(JavaScript側での関数名)
    - 呼び出す関数のポインタ(myfunc)
    - 引数の数
  - JS_DefineFunctions()で、JSFunctionSpec構造体を登録

module.jp

拡張例 - save_to_file()

module.jp

拡張例 - save_to_file()

Terminal.app — vim — ⌘1

```
#include <stdio.h>
#include <jsapi.h>


static JSBool my_save_to_file(JSContext *ctx, JSObject *obj,
                              uintN argc, jsval *argv, jsval *rval)
{
    const char *fname, *text;
    FILE *fp;

    fname = JS_GetStringBytes(JS_ValueToString(ctx, argv[0]));
    text = JS_GetStringBytes(JS_ValueToString(ctx, argv[1]));

    fp = fopen(fname, "w");
    fprintf(fp, text);

    return JS_TRUE;
}

static JSFunctionSpec my_functions[] = {
    { "save_to_file", my_save_to_file, 2, 0, 0 },
    { 0, 0, 0, 0, 0 },
};
"save.c" 115L, 2700C written                          1,1           Top
```

module.jp

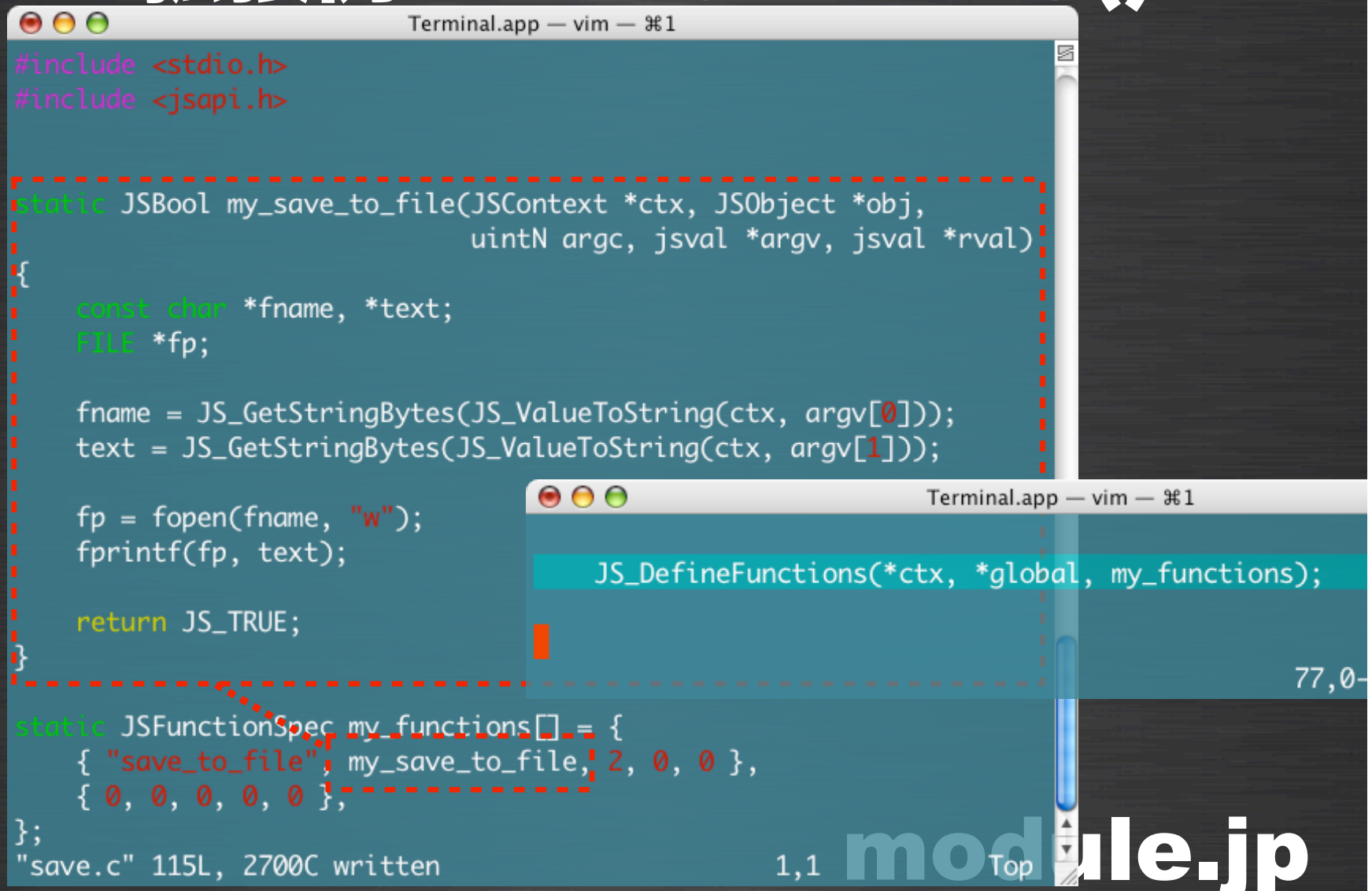
拡張例 - save_to_file()

Terminal.app — vim — ⌘1

```
#include <stdio.h>
#include <jsapi.h>


static JSBool my_save_to_file(JSContext *ctx, JSObject *obj,
                              uintN argc, jsval *argv, jsval *rval)
{
    const char *fname, *text;
    FILE *fp;

    fname = JS_GetStringBytes(JS_ValueToString(ctx, argv[0]));
    text = JS_GetStringBytes(JS_ValueToString(ctx, argv[1]));

    fp = fopen(fname, "w");
    fprintf(fp, text);

    return JS_TRUE;
}

static JSFunctionSpec my_functions[] = {
    { "save_to_file", my_save_to_file, 2, 0, 0 },
    { 0, 0, 0, 0, 0 },
};
"save.c" 115L, 2700C written                          1,1           Top
```

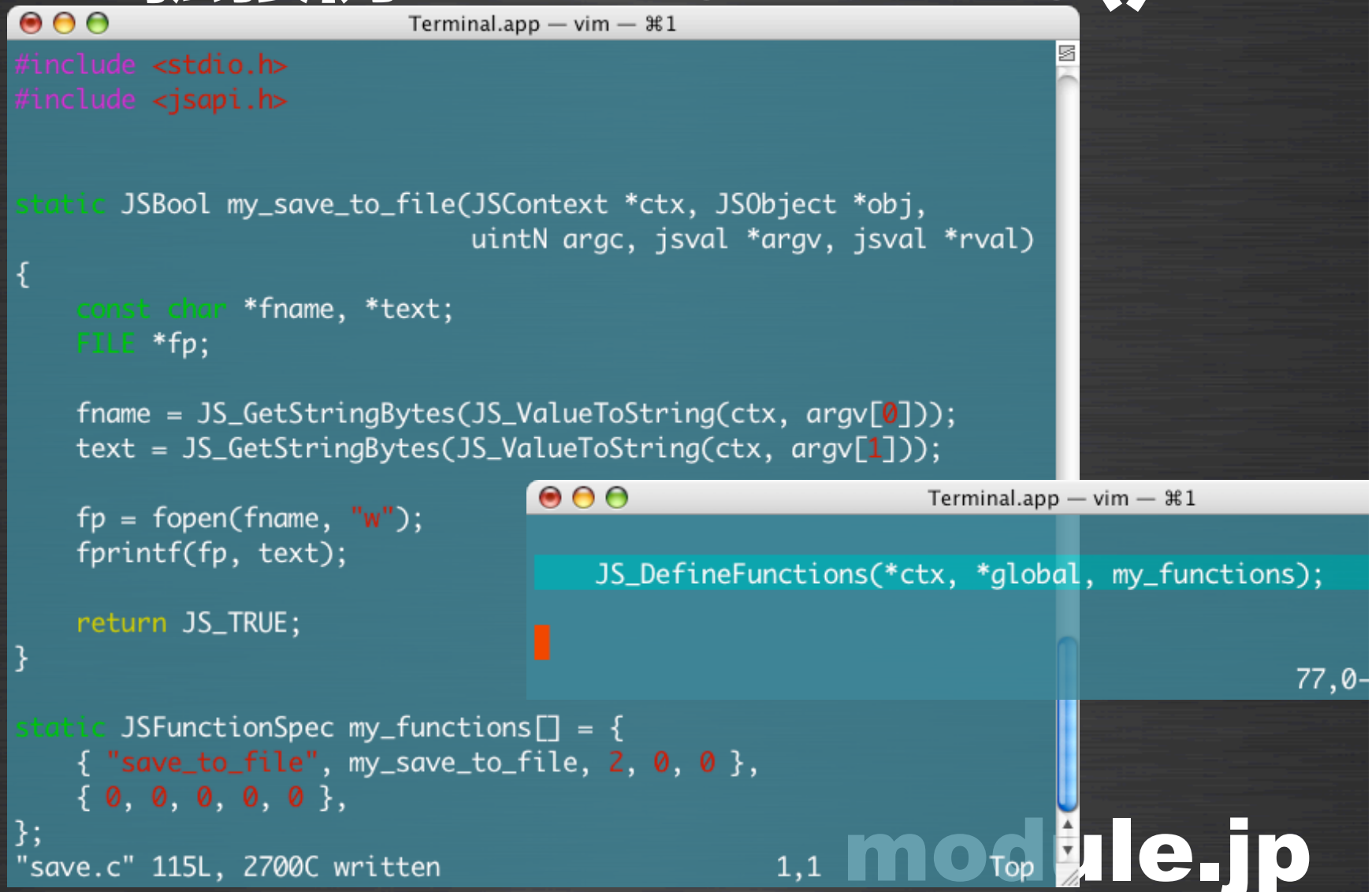
module.jp

拡張例 - save_to_file()



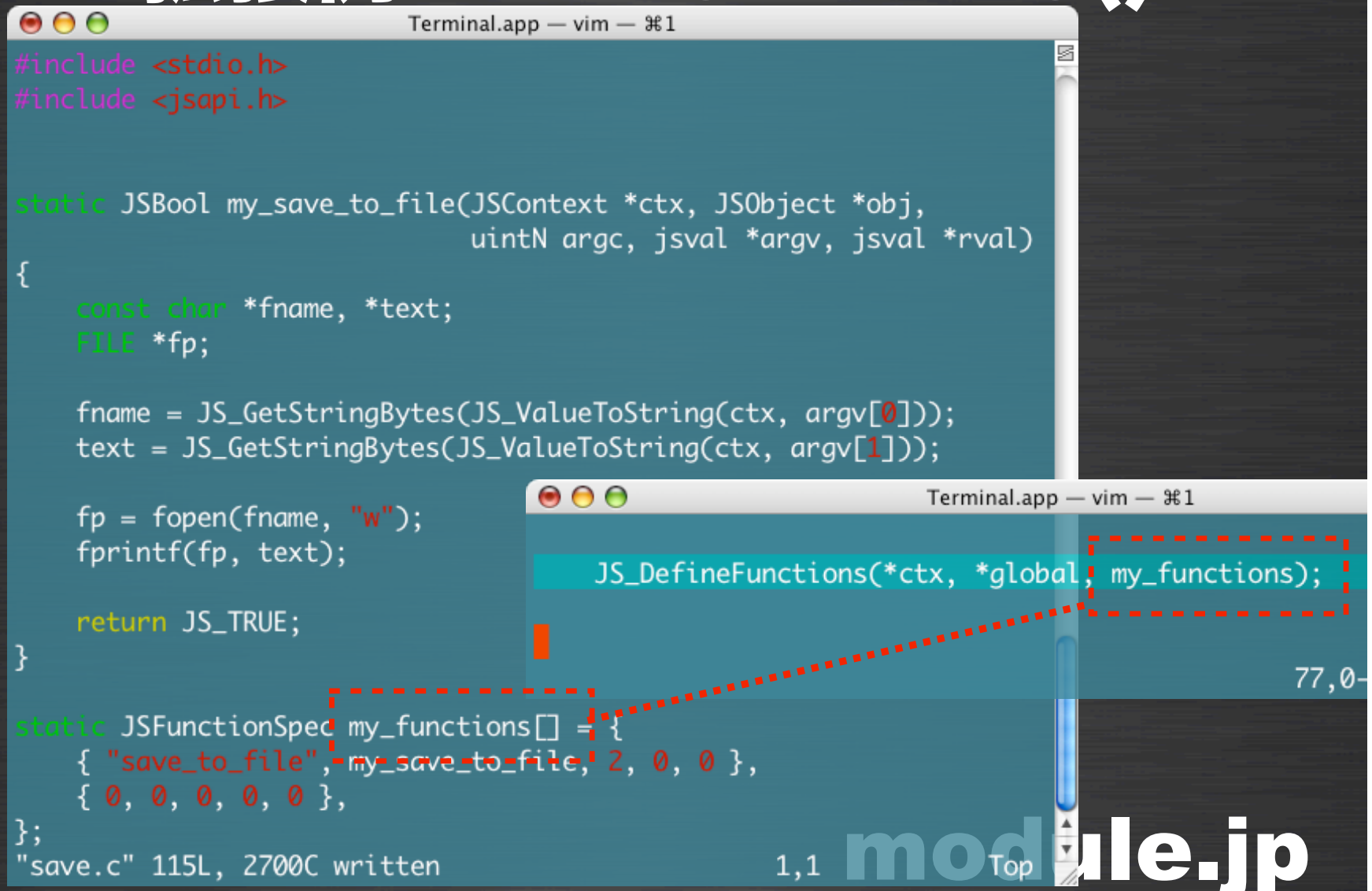
module.jp

拡張例 - save_to_file()



module.jp

拡張例 - save_to_file()

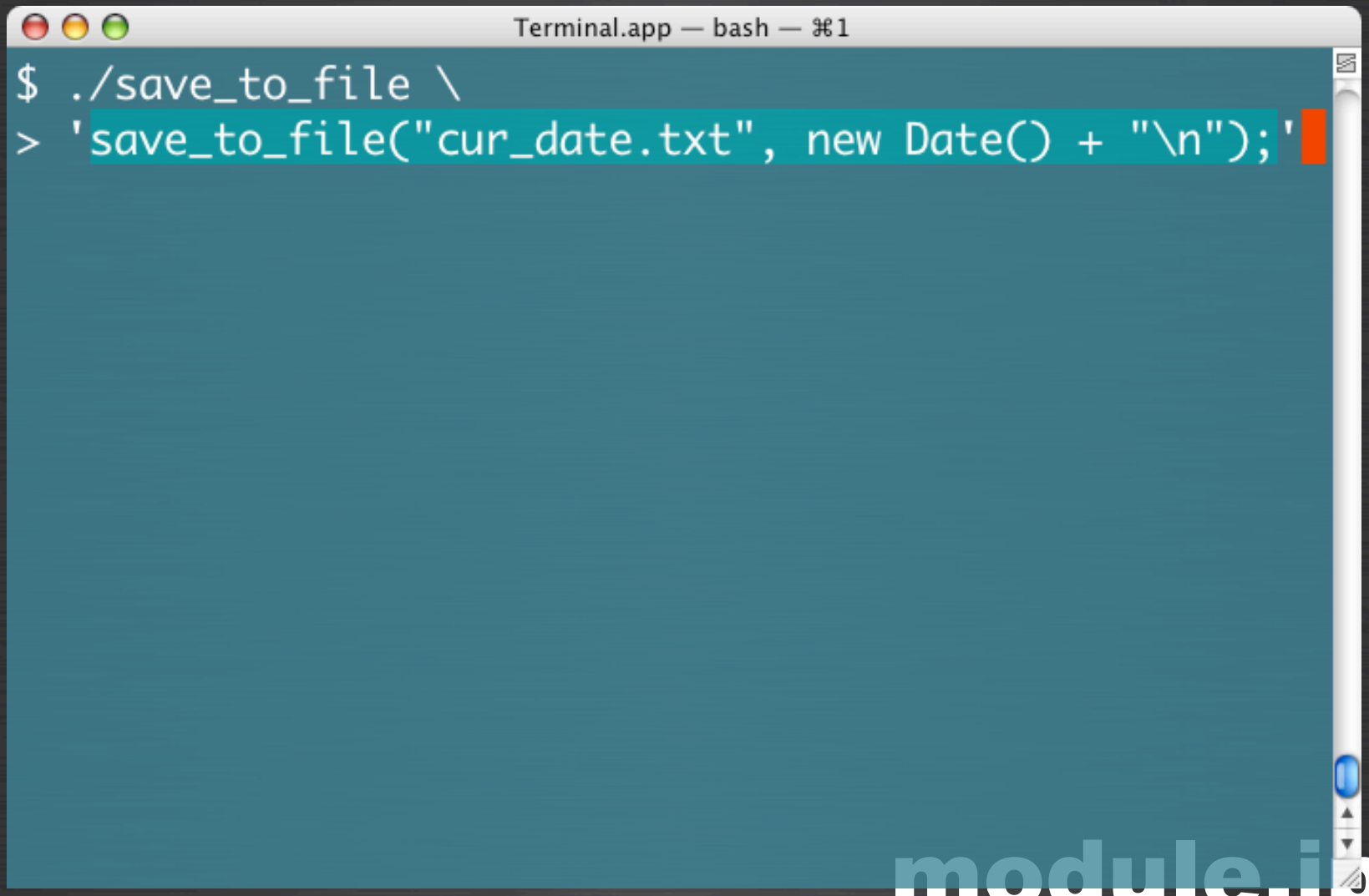


module.jp

拡張例 - save_to_file()

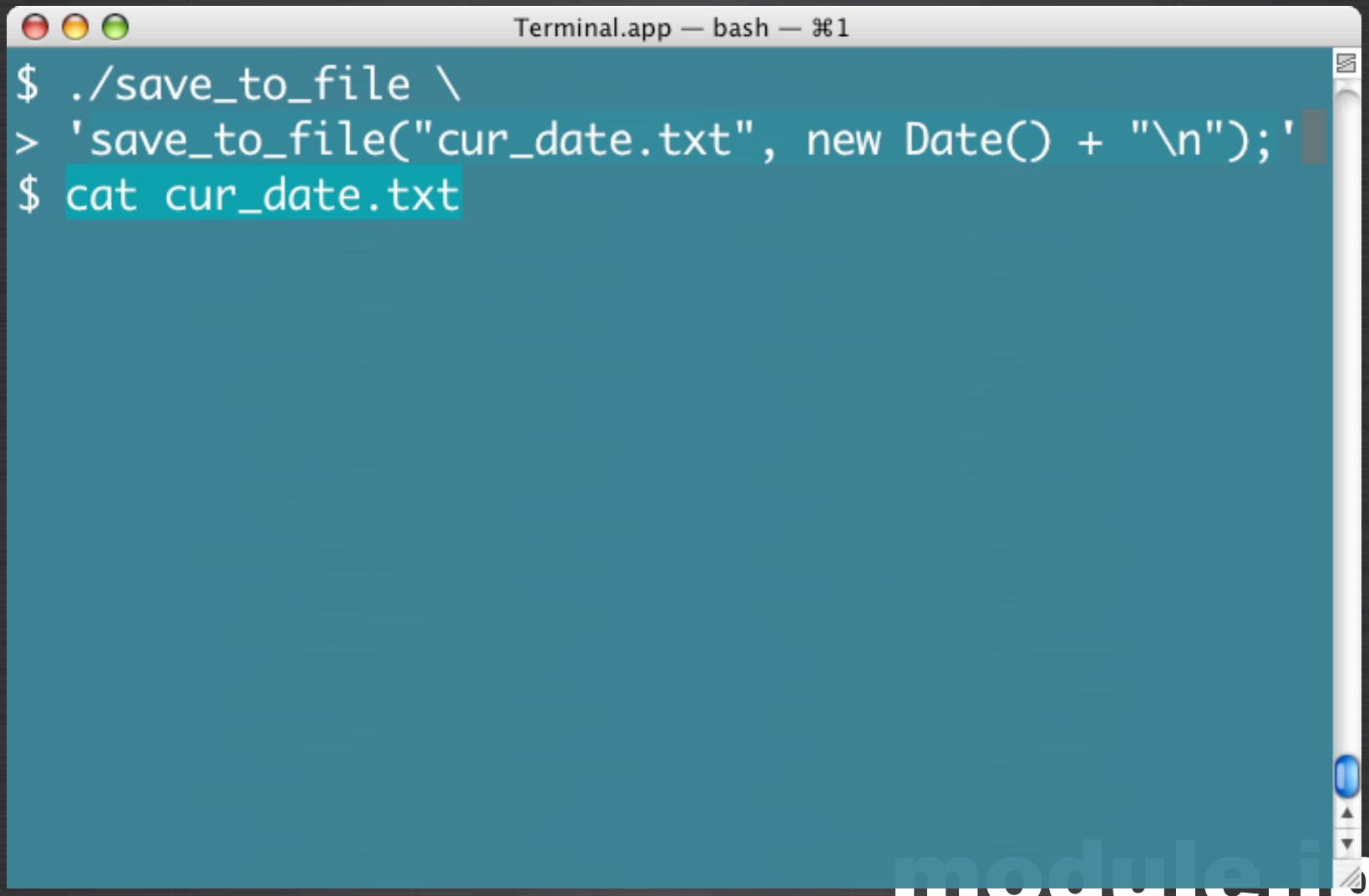
module.jp

拡張例 - save_to_file()



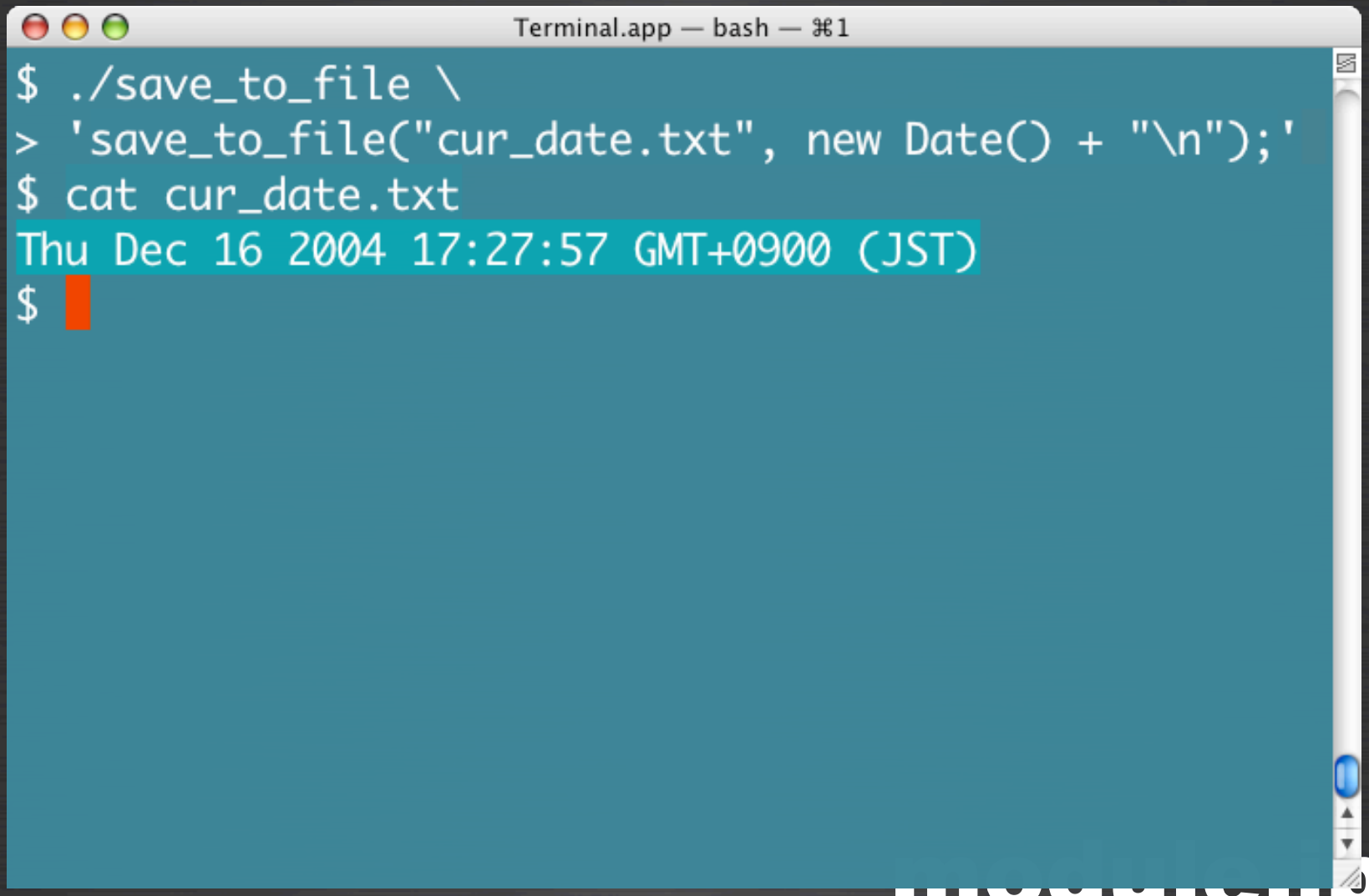
module.jp

拡張例 - save_to_file()



module.jp

拡張例 - save_to_file()



利用例

- Apache moduleでPHPみたいなやつ
- HTML側にJavaScriptで記述したvalidationロジックを、サーバ側で実行・検証する。
- アプリケーションのカスタマイズ用埋込言語

module.jp

再考してみると?

- ナニゲに面白いぞJavaScript
  - JavaScript悪くない。悪いはブラウザ
- イジリ甲斐があるぞJavaScript
  - OSSの処理系が手に入るぞJavaScript
  - 拡張し甲斐があるぞJavaScript
- 組み込みたくなっちゃうぞJavaScript
  - 書ける人 or 書けそうな人って多いよねJavaScript

module.jp

再考ですかーっ！？

JavaScript

module.jp

module.jp

最高ですかーっ！？

JavaScript

module.jp

小山浩之 Hiroyuki OYAMA
〒177-0053
東京都練馬区関町南4-17-1 106
Mail: oyama@module.jp
Web: http://module.jp/

module.jp