2. REPL（リプル）

REPL（リプル）とは

REPL(Read-Eval-Print Loop) は、ユーザーとインタプリタが対話的にコードを実行して、入力、評価、出力の繰り返す作業ことを指します。 インタプリタ とは、コンピュータが実行できるようにコード変換しながら同時に実行していくソフトウェアです。 Node.js もREPLでJavaScriptを実行できます。

Node.js で REPL を実行する

REPL の開始

Node.js の RELP を利用するには、nodeコマンドで開始します。

```
$ node
Welcome to Node.js v12.18.2.
```

プログラムの入力

ターミナルにプログラムを入力して実行します。

```
> 2+3
5
> 3*1.5
4.5
> x = 5
5
> y = 2
2
> x / y
2.5
```

REPL の終了

`.exit` で `REPL` を終了します。

```
> .exit
```

REPLのコマンド

`REPL` の主なコマンドは以下のとおりです。

コマンド	動作
Ctrl + C	現在のターミナル終了
Ctrl + C（2回）	終了
Ctrl + D	終了（ターミナルの終了と同じ）
Tab	コマンドを一覧表示
.help	入力コマンドのヘルプ表示
.save	ファイル保存
.load	ファイル読み込み

node コマンド実行

`hello.js` ファイルを作成して、 `console.log()` を記述し、**nodeコマンドでJavaScriptを実行**してみましょう。

```
"use strict";

console.log("Hello Node!");
```

use strict

`use strict` は、プログラムを `strict` （厳格）モードで実行するモードで、**エラーチェックが厳しく**なります。ターミナルで `node` コマンドを実行すると、**JavaScriptの実行結果が表示**されます。

```
% node hello.js
Hello Node!
```

ファイル読み込み

`.load` コマンドでファイルを読み込みます。

```
.load ファイル名
```

`hello.js` を実行してみます。

```
% node
> .load hello.js
console.log('Hello Node!');
Hello Node!
undefined
```

ファイル保存

プログラムを1行ずつ実行して `.save` でファイル保存できます。

```
.save ファイル名
```

プログラムを入寮して `hello.js` に保存してみます。

```
> let message = 'Hello Tokyo'
'Hello Tokyo'
> console.log(message)
Hello Tokyo
undefined
> .save hello.js
Session saved to: hello.js
> .exit
```

プログラム追記の確認

hello.js にプログラムが追記されているか確認してみましょう。

```
"use strict";

console.log("Hello Node!");

message= 'Hello Tokyo'
console.log(message)
```

リテラル式

JavaScript で文字列と変数を同時に記述することで、コードがシンプルになります。リテラル式 を利用すると**文字列の中に変数を入れて記述**できます。リテラル式は、 `(バッククオート) で囲んで変数と文字列を記述します。

```
`${変数名}`
```

hello2.js を作成して以下のコードを記述します。

```
let name = 'Tokyo'

console.log(`Hello ${name}`)
```

リテラル式の %s で変数代入

文字列の中に %s を利用して、変数を代入することもできます。hello2.js に追記してみましょう。

```
console.log("Hello %s", name);
```

hello2.js を実行してみましょう。

```
% node hello2.js
Hello Tokyo
Hello Tokyo
```

ファイルの追記保存

REPL で hello.js にプログラムを追記して保存します。

```
% node
> .load hello.js
> message = 'Hello Tokyo'
'Hello Tokyo'
> console.log(message)
Hello Tokyo
undefined
> .save hello.js
Session saved to: hello.js
> .exit
```

message.js の作成

`message.js` を作成します。

```
"use strict";

let messages = [ "Apple", "Orange", "Peach" ];
```

message.js の読み込みと利用

`REPL` で `message.js` を読み込みます。

```
$ node
> .load message.js
"use strict";

let messages = [ "Apple", "Orange", "Peach" ];
'use strict'
```

`messages` をコンソール表示します。

```
> console.log(messages)
[ 'Apple', 'Orange', 'Peach' ]
```

`forEach()` で `messages` を表示するプログラムを入力します。

```
> messages.forEach(message => console.log(message))
Apple
Orange
Peach
undefined
```

outMessage.js ファイルに保存

`outMessage.js` ファイルに保存して、 `REPL` を終了します。

```
> messages.forEach(message => console.log(message))
Apple
Orange
Peach
undefined
> .exit
```

outMessage.js ファイルの確認

保存した `outMessage.js` ファイルの中身を確認してみましょう。

```
"use strict";

let messages = [ "Apple", "Orange", "Peach" ];
console.log(messages)
messages.forEach(message => console.log(message))
```