

# IBM i World 2022

IBM i コンテンツ (2022年4月版)

## Node.jsでWebアプリを作ってみよう

日本アイ・ビー・エム株式会社  
テクノロジー事業本部  
IBM Powerテクニカルセールス  
澤田英寿

# 目次

1. Node.js 概説
2. Node.js のインストール手順
  - 1) sshdの開始
  - 2) yumの導入
  - 3) Node.jsの導入と確認
3. Node.jsでWebアプリを作ってみよう
  - 1) Hello.jsを作ってみよう
  - 2) 既存のデータベースにアクセスしてみよう
  - 3) itoolkitを使ってIBM i コマンドをWebでアクセスしてみよう
4. 補足情報

# 1. Node.js 概説

2009年リリースのオープンソース



非同期型のイベント駆動モデルを採用

Web業界で広く使われているJava Script環境

世界で最も大きなオープンソースのライブラリーエコシステム

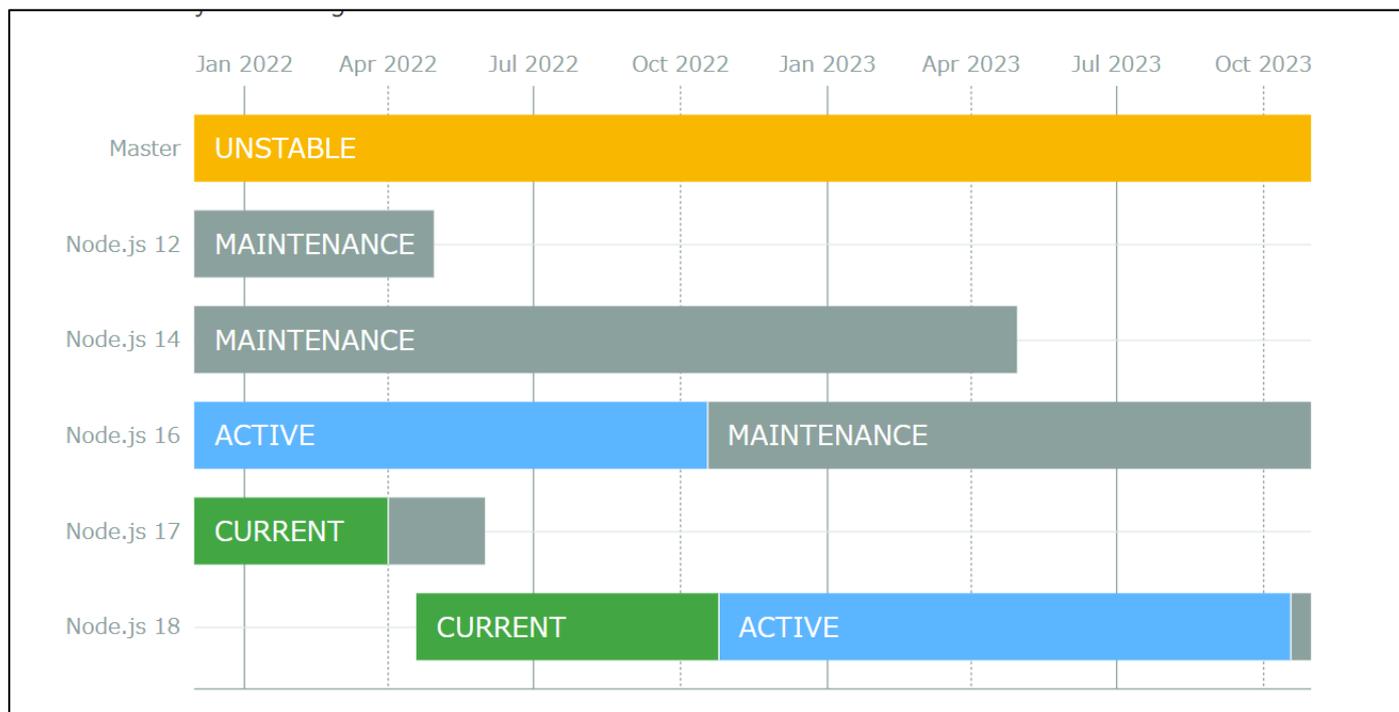
IBM i にはIBMがポーティング

IBM i リソースへのアクセス機能のAPIセットも提供

- ・ Node.js v12
- ・ Node.js v14

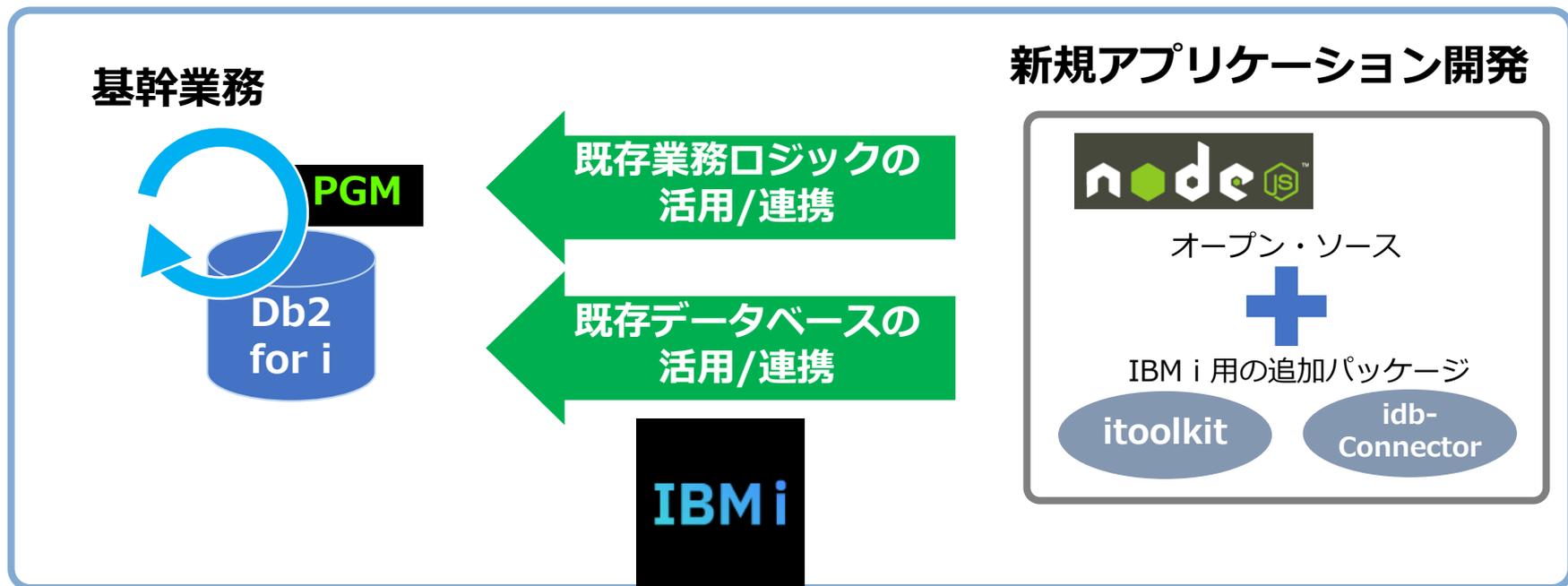
## (参考) Node.js リリース・スケジュール情報

参照 : <https://github.com/nodejs/Release>



# レガシーアプリケーション資産とNode.js連携

- 1) PGMをはじめとしたシステム資源にアクセスするための**itoolkit**
- 2) 基幹データ (Db2 for i) を操作するための**idb-Connector**



## (参考) アプリケーション資産とNode.js連携

### 1) PGMをはじめとしたシステム資源にアクセスするための**itoolkit**

本資料では、「**itoolkit**」を使った簡単なプログラムをご紹介します。

itoolkitは、IBM i のデータベース以外の IBM i システム資源へのアクセスを提供

- ・ コマンド
- ・ プログラム
- ・ システム状況
- ・ ジョブ情報
- ・ ユーザー空間
- ・ データ待ち行列
- ・ オブジェクト情報
- ・ . . .

詳細は、下記を参照してください。

<https://www.npmjs.com/package/itoolkit>

### 2) 基幹データ (Db2 for i) を操作するための**idb-Connector**

本資料では、「**idb-connector**」を使った簡単なプログラムをご紹介します。

idb-connectorは、DB2 for i データベース・オブジェクトへのアクセスを提供

詳細は、下記を参照してください。

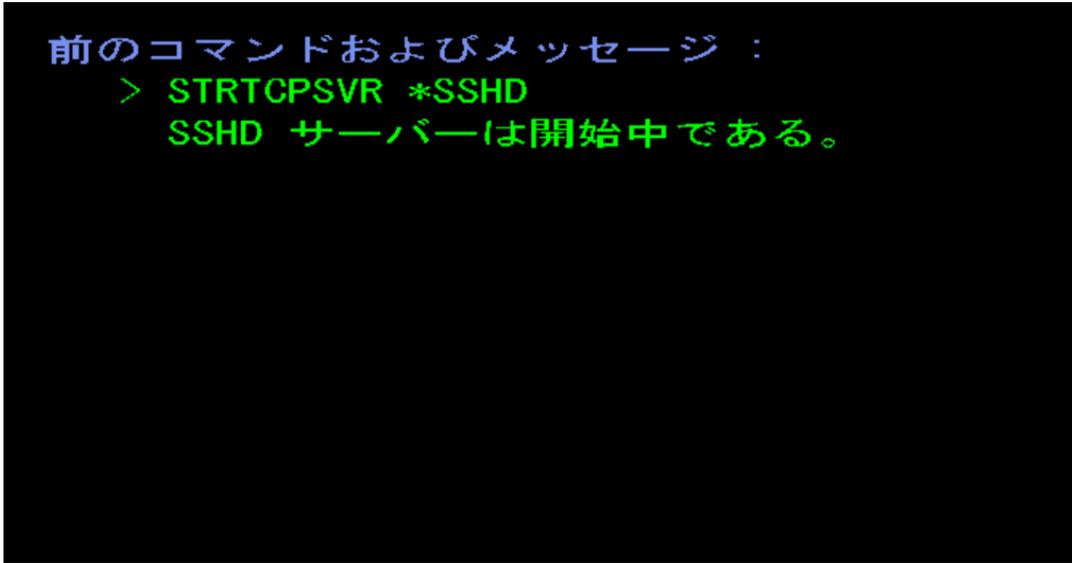
<https://www.npmjs.com/package/idb-connector>

## 2. Node.js のインストール手順

### (1) sshdの開始

5250エミュレーターで、以下のCLコマンドを実行してください。

```
STRTCPSVR *SSHD
```



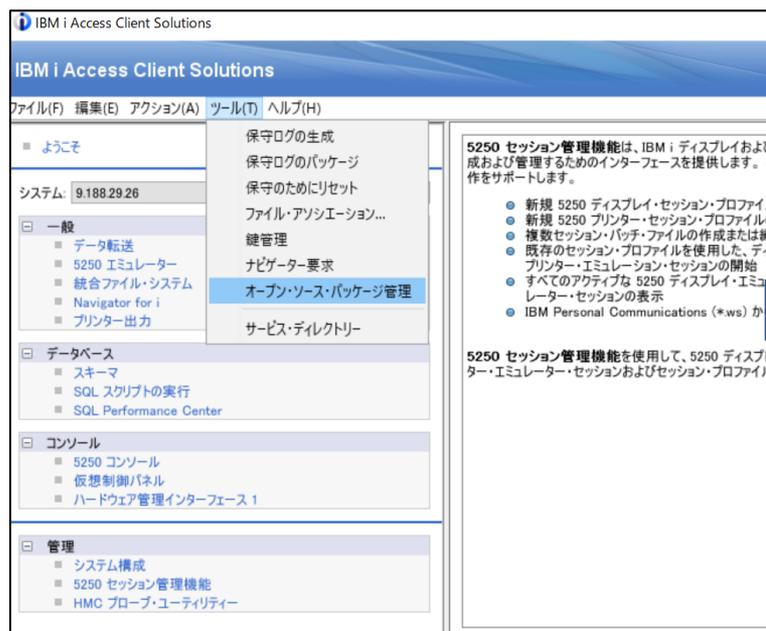
```
前のコマンドおよびメッセージ :  
> STRTCPSVR *SSHD  
SSHD サーバーは開始中である。
```

## 2. Node.js のインストール手順

### (2) yumの導入-1

yumについての詳細は、補足資料のガイドをご覧ください。  
ここでは簡単な手順のみ解説します。

ACSの「オープンソースパッケージ管理」を選択

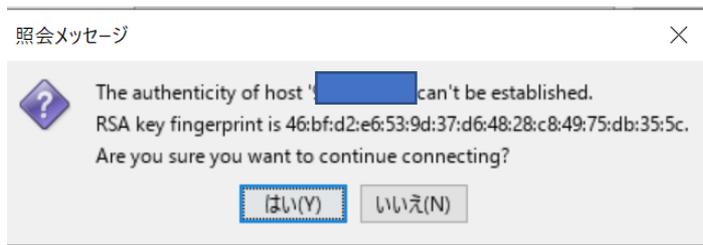


システム名、ユーザーID、パスワードを指定

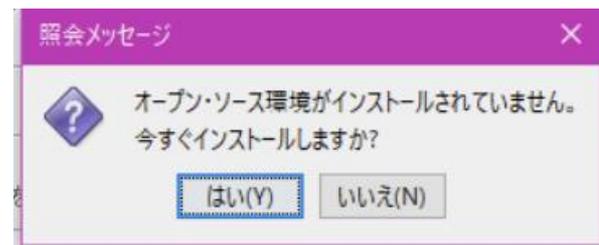
## 2. Node.js のインストール手順

### (2) yumの導入-2

RSAキーの警告がでますが、「はい」を選択



下記のメッセージで「はい」を選択

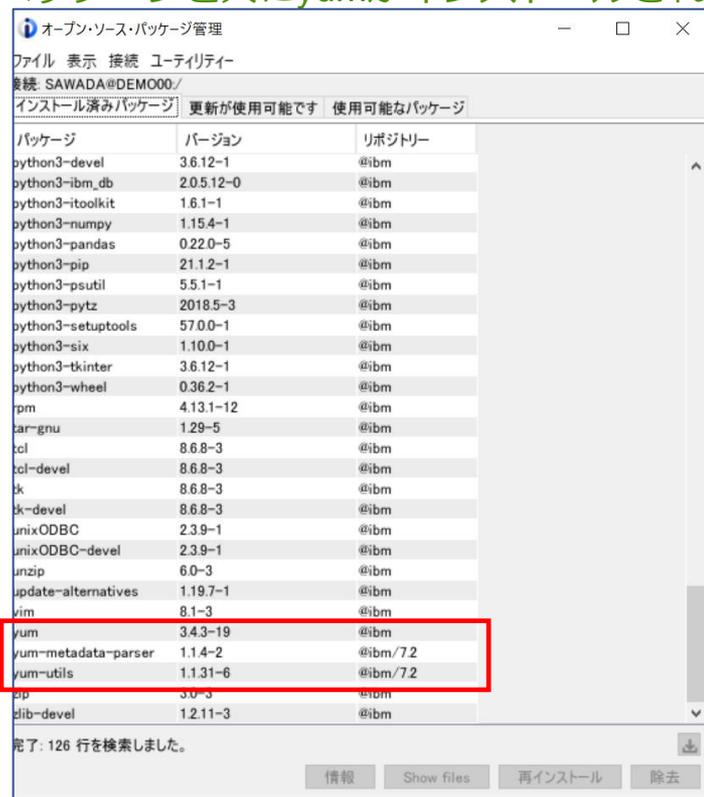


## 2. Node.js のインストール手順

### (2) yumの導入-3

導入が完了すると、以下のようなOSSパッケージが表示されます。

- ・基本的なパッケージと共にyumがインストールされます。



## 2. Node.js のインストール手順

### (3) Node.jsの導入-1

先ほど導入したyumとACSを利用してNode.jsを導入します。

ACSのオープンソースパッケージ管理から  
「使用可能なパッケージ」タブを選択する。

オープンソースパッケージ管理

ファイル 表示 接続 ユーティリティ

接続: [ ]

インストール済みパッケージ 更新が使用可能です **使用可能なパッケージ**

パッケージ	バージョン	リポジトリ
ncurses-terminfo	6.0-8	ibm
ncurses-tools	6.0-8	ibm
nginx	1.16.1-4	ibm
ninja-build	1.10.1-1	ibm
nodejs	8.9.3-0	ibm
nodejs10	10.24.1-3	ibm
nodejs12	12.22.9-1	ibm
nodejs14	14.18.3-1	ibm
nodejs8	8.17.0-2	ibm
nodever	1.0.0-4	ibm
npth-devel	1.6-0	ibm
nspr	4.32-1	ibm
nspr-devel	4.32-1	ibm
nss	3.68.1-1	ibm
nss-devel	3.68.1-1	ibm
oniguruma-devel	6.9.4-1	ibm
openjdk-11-ea	11.0.11.9-1	ibm
openjdk-11-jmods-ea	11.0.11.9-1	ibm
openjdk-11-src-ea	11.0.11.9-1	ibm
openssh	8.1p1-1	ibm
openssh-server	8.1p1-1	ibm
openssl	1.1.1k-1	ibm
openssl-devel	1.1.1k-1	ibm
openssl-engines	1.1.1k-1	ibm
p11-kit	0.23.14-2	ibm
p11-kit-devel	0.23.14-2	ibm
p11-kit-trust	0.23.14-2	ibm
p7zip	16.02-2	ibm

完了: 514 行を検索しました。

Node.js14を選択し、  
「インストール」を選択する。

インストール済みパッケージ 更新が使用可能です 使用可能なパッケージ

パッケージ	バージョン	リポジトリ
ncurses-terminfo	6.0-8	ibm
ncurses-tools	6.0-8	ibm
nginx	1.16.1-4	ibm
ninja-build	1.10.1-1	ibm
nodejs	8.9.3-0	ibm
nodejs10	10.24.1-3	ibm
nodejs12	12.22.9-1	ibm
<b>nodejs14</b>	<b>14.18.3-1</b>	<b>ibm</b>
nodejs8	8.17.0-2	ibm
nodever	1.0.0-4	ibm
npth-devel	1.6-0	ibm
nspr	4.32-1	ibm
nspr-devel	4.32-1	ibm
nss	3.68.1-1	ibm
nss-devel	3.68.1-1	ibm
oniguruma-devel	6.9.4-1	ibm
openjdk-11-ea	11.0.11.9-1	ibm
openjdk-11-jmods-ea	11.0.11.9-1	ibm
openjdk-11-src-ea	11.0.11.9-1	ibm
openssh	8.1p1-1	ibm
openssh-server	8.1p1-1	ibm
openssl	1.1.1k-1	ibm
openssl-devel	1.1.1k-1	ibm
openssl-engines	1.1.1k-1	ibm
p11-kit	0.23.14-2	ibm
p11-kit-devel	0.23.14-2	ibm
p11-kit-trust	0.23.14-2	ibm
p7zip	16.02-2	ibm

完了: 514 行を検索しました。

情報 **インストール**

## 2. Node.js のインストール手順

### (3) Node.jsの導入-2

下記の画面が表示されるので、  
「y」を選択する。

```
Package Installation
Could not chdir to home directory /home/SAWADA: No such file or directory
Setting up Install Process
Resolving Dependencies
--> Running transaction check
--> Package nodejs14.ppc64 0:14.18.3-1 will be installed
--> Processing Dependency: /QOpenSys/pkgs/bin/python3 for package: nodejs14-14.18.3-1.ppc64
--> Processing Dependency: python3 for package: nodejs14-14.18.3-1.ppc64
--> Running transaction check
--> Package python3.ppc64 0:3.6.15-1 will be installed
--> Processing Dependency: lib:/QOpenSys/pkgs/lib/libreadline.so.8(shr_64.o)(ppc64) for package: python3-3.6.15
--> Running transaction check
--> Package libreadline8.ppc64 0:8.1-2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch      Version      Repository    Size
=====
Installing:
nodejs14          ppc64    14.18.3-1    ibm           39 M
Installing for dependencies:
libreadline8     ppc64    8.1-2        ibm           533 k
python3          ppc64    3.6.15-1    ibm           36 M
=====
Transaction Summary
-----
Install      3 Packages

Total download size: 76 M
Installed size: 273 M
Is this ok [y/N]: y
```

Node.jsがインストールされたら  
下記のようなメッセージがでます。これで  
OKです。

```
Installing for dependencies:
libreadline8     ppc64    8.1-2        ibm           533 k
python3          ppc64    3.6.15-1    ibm           36 M
-----
Transaction Summary
-----
Install      3 Packages

Total download size: 76 M
Installed size: 273 M
Is this ok [y/N]: y
Downloading Packages:
(1/3): libreadline8-8.1-2.1b1i7.2.ppc64.rpm | 533 kB 00:01
(2/3): nodejs14-14.18.3-1.1b1i7.2.ppc64.rpm | 39 MB 01:45
(3/3): python3-3.6.15-1.1b1i7.2.ppc64.rpm | 36 MB 01:39
-----
Total                                           372 kB/s | 76 MB 03:28
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing: libreadline8-8.1-2.ppc64                1/3
  Installing: python3-3.6.15-1.ppc64                 2/3
update-alternatives: using /QOpenSys/pkgs/bin/python3.6 to provide /QOpenSys/pkgs/bin/python3 (python3) in auto
update-alternatives: using /QOpenSys/pkgs/bin/python3.6 to provide /QOpenSys/pkgs/bin/python (python) in auto
  Installing: nodejs14-14.18.3-1.ppc64                3/3
update-alternatives: using /QOpenSys/pkgs/lib/nodejs14/bin/node to provide /QOpenSys/pkgs/bin/node (node) in auto
-----
Installed:
nodejs14.ppc64 0:14.18.3-1

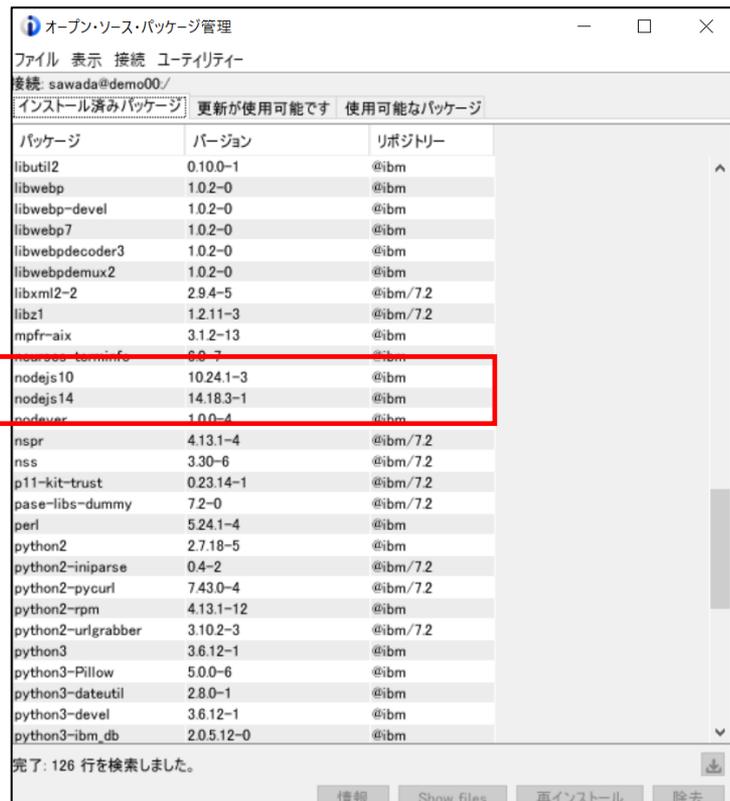
Dependency Installed:
libreadline8.ppc64 0:8.1-2      python3.ppc64 0:3.6.15-1

Complete!
```

## 2. Node.js のインストール手順

### (3) Node.jsの導入-3

Node.jsのインストールが完了すると、下記のように、「インストール済みパッケージ」にNode.jsが表示されます。このマシンにはNode.js v10とv14が導入済みです。



オープン・ソース・パッケージ管理

ファイル 表示 接続 ユーティリティ

接続: sawada@demo00/

インストール済みパッケージ | 更新が使用可能です | 使用可能なパッケージ

パッケージ	バージョン	リポジトリ
libutil2	0.10.0-1	@ibm
libwebp	1.0.2-0	@ibm
libwebp-devel	1.0.2-0	@ibm
libwebp7	1.0.2-0	@ibm
libwebpdecoder3	1.0.2-0	@ibm
libwebpdemux2	1.0.2-0	@ibm
libxml2-2	2.9.4-5	@ibm/7.2
libz1	1.2.11-3	@ibm/7.2
mpfr-aiix	3.1.2-13	@ibm
nodejs-bininfo	6.0.7	@ibm
nodejs10	10.24.1-3	@ibm
nodejs14	14.18.3-1	@ibm
nodever	1.0.0-4	@ibm
nspr	4.13.1-4	@ibm/7.2
nss	3.30-6	@ibm/7.2
p11-kit-trust	0.23.14-1	@ibm/7.2
pase-libs-dummy	7.2-0	@ibm/7.2
perl	5.24.1-4	@ibm
python2	2.7.18-5	@ibm
python2-iniparse	0.4-2	@ibm/7.2
python2-pycurl	7.43.0-4	@ibm/7.2
python2-rpm	4.13.1-12	@ibm
python2-urlgrabber	3.10.2-3	@ibm/7.2
python3	3.6.12-1	@ibm
python3-Pillow	5.0.0-6	@ibm
python3-dateutil	2.8.0-1	@ibm
python3-devel	3.6.12-1	@ibm
python3-ibm_db	2.0.5.12-0	@ibm

完了: 126 行を検索しました。

情報 Show files 再インストール 除去

## 2. Node.js のインストール手順

### (3) Node.jsの導入確認 - 1

Node.jsは、他の多くのオープンソーステクノロジーと同様に、IBM iのPASE環境で実行されます。

Node.jsが正しくインストールされていることを確認してみましょう。5250は、英語小文字が表示できる設定（ホストコードページ939など）にしておきます。

5250画面から、下記のコマンドでPASE環境に入る。（「call qp2term」コマンド）

下記の画面になります。

```
MAIN                               IBM I   メインメニュー
次の 1 つを選択してください。

1. ユーザー・タスク
2. オフィス・タスク
3. 汎用システム・タスク
4. ファイル、ライブラリー、およびフォルダー
5. プログラミング
6. 通信
7. システムの定義または変更
8. 問題処理
9. メニューの表示
10. 情報援助オプション
11. IBM I ACCESS のタスク

90. サインオフ

選択項目またはコマンド
===> call qp2term
```



```
/Q0penSys/usr/bin/-sh
$
===> _
```

## 2. Node.js のインストール手順

### (3) Node.jsの導入確認 - 2

yumでインストールされたOSSは、IFS上の /Opensys/pkgs/binの下に導入されますので、環境変数PATHを変更しておきます。

下記のコマンドを使用してPATHを変更します

```
export PATH=/QOpenSys/pkgs/bin:$PATH
```

```
$  
> export PATH=/QOpenSys/pkgs/bin:$PATH
```



「Node -v 」コマンドで、導入されているNode.jsのバージョン (v14.18.3)が確認できました。これで導入確認は完了です。

```
> node -v  
v14.18.3  
$  
===> _____
```

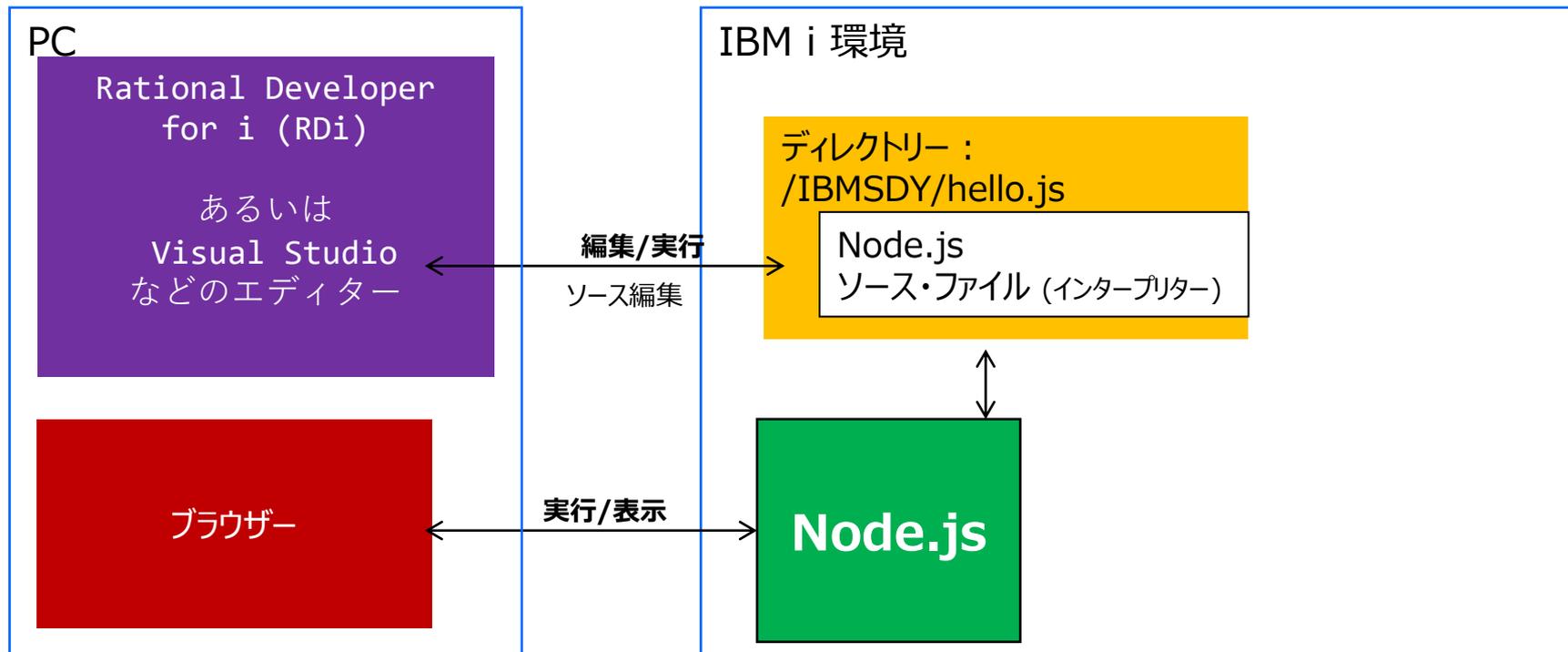
### 3. Node.jsでWebアプリを作ってみよう

#### 1) Hello.jsを作ってみよう（実行環境イメージ）

Node.jsのソース編集をRational Developer for i (RDi)で行います。

RSE (Remote System Explorer) を使用して、IBM iのIFSのソースファイルを直接編集します。

他の使い慣れたエディターでもOKです\*。



### 3. Node.jsでWebアプリを作ってみよう

#### 1) Hello.jsを作ってみよう-1 (ソースコードの作成)

まず5250画面で、IFS上に、IBMSDYを作成します。

```
コマンドを入力して、実行
==> MD DIR(' /IBMSDY')
```



RDİを起動して、IBM iに接続してください。

- ・RDİで、フィルターを設定します。

(IBMSDYのみの表示になるようにします)

下記のように、「IFSファイル」→「新規」→「フィルター」を選択



下記のように設定して、「次へ」を選択



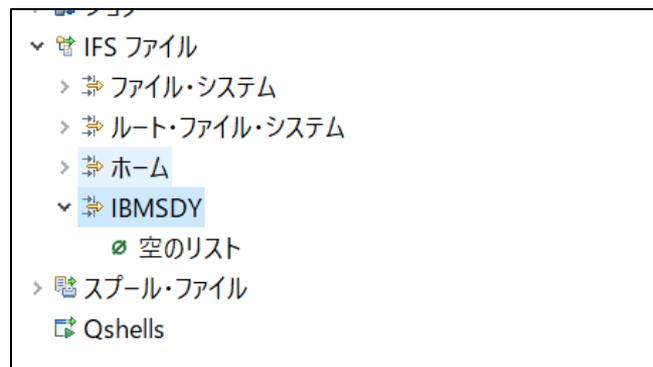
## 3. Node.jsでWebアプリを作ってみよう

### 1) Hello.jsを作ってみよう-2 (ソースコードの作成)

下記の画面で、「終了」を選択



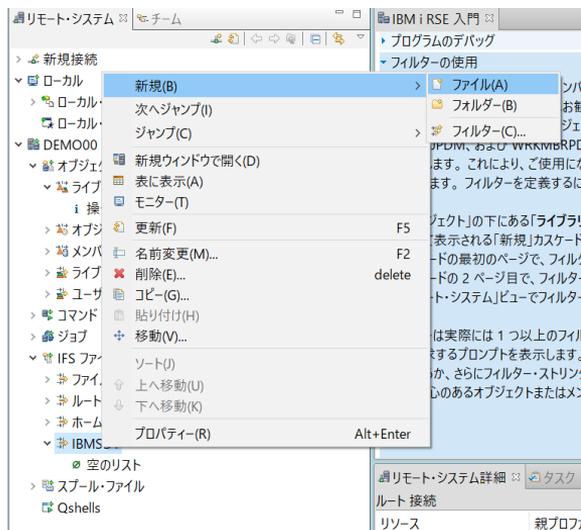
IFSファイルのリストに、IBMSDYが表示されればOKです



# 3. Node.jsでWebアプリを作ってみよう

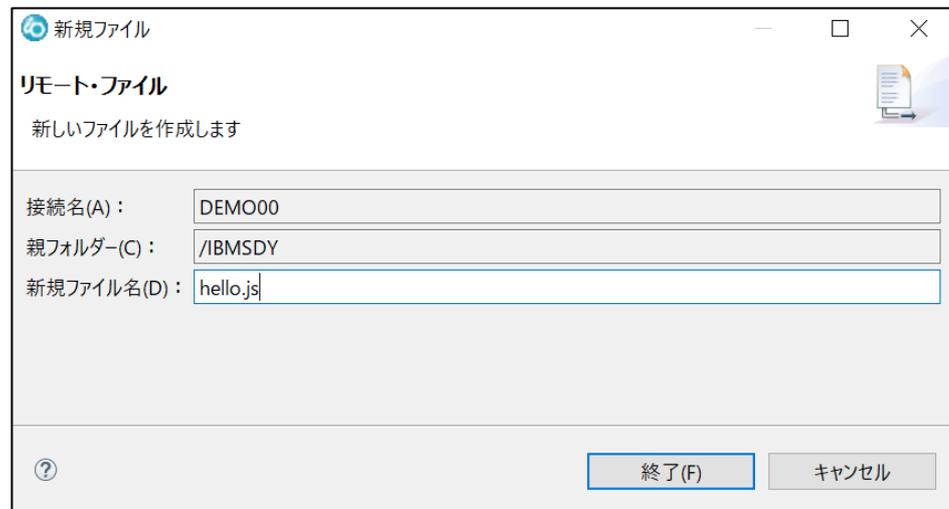
## 1) Hello.jsを作ってみよう-3 (ソースコードの作成)

作成済みの IFS フィルター「IBMSDY」を右クリック、メニューから「新規」→「ファイル」をクリックします



以下のように入力し、終了を選択。

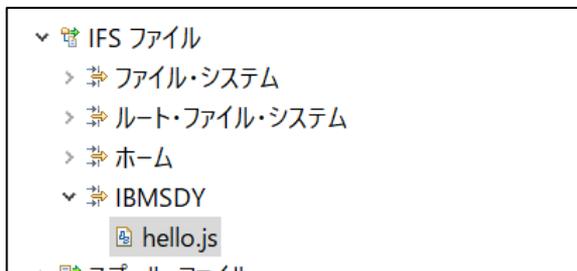
新規ファイル名 : hello.js



## 3. Node.jsでWebアプリを作ってみよう

### 1) Hello.jsを作ってみよう-4 (ソースコードの作成)

IFS 上に **hello.js** が作成されます、  
ダブルクリックしエディターを開きます



下記のようにソースコードを入力し、保存します

このソースコードは、ブラウザに 'hello world' を表示し終了するものです。

```
001 var http = require('http');
002 var ServerPort = 59999;
003 var server = http.createServer(function (req, res) {
004   res.writeHead(200, {'Content-Type': 'text/plain; charset=utf-8'});
005   res.write('Hello World!! by Node.js');
006   res.end();
007 });
008 server.listen(ServerPort);
009 console.log('run server port = '+ ServerPort);
```

### 3. Node.jsでWebアプリを作ってみよう

#### 1) Hello.jsを作ってみよう-5 (プログラムの実行)

Node.jsの導入の確認で実行したPASE環境を起動します。[call qp2term] コマンド

```
コマンドを入力して、実行キーを  
==> call qp2term_
```



Node.jsの導入の確認で実行したPATHを指定します

```
$  
> export PATH=/QOpenSys/pkgsrc/bin:$PATH  
$
```

以下のコマンドでNode.jsを起動します。  
Node.jsが起動すると、下記のように、ブラウザーで指定するポート番号が表示されます。

-> node /IBMSDY/hello.js を打鍵します



```
> node /IBMSDY/hello.js  
run server port = 59999
```

### 3. Node.jsでWebアプリを作ってみよう

#### 1) Hello.jsを作ってみよう-6 (プログラムの実行)

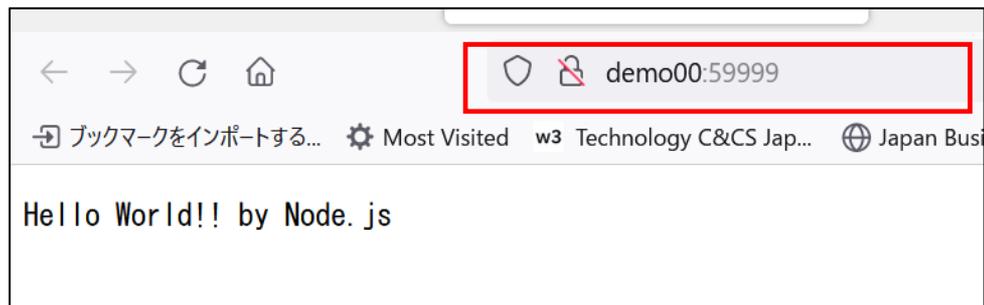
PCでブラウザを起動し、以下のURLを実行します。

XXX.XXX.XXX.XXXは、IBM iのIPアドレス(又はホスト名)を指定します。

`http://xxx.xxx.xxx.xxx:59999/`

ここではホスト名：DEMO00マシンのポート：59999を指定しました。

- 成功すると、下記のようにブラウザに「Hello World!! by Node.js」が表示されます。



## 3. Node.jsでWebアプリを作ってみよう

### 1) Hello.jsを作ってみよう-7 (プログラムの解説1)

簡単にプログラムを解説します。

#### HTTP サーバーの作成

※ HTTP サーバーとして動作する箇所の説明です

#### HTTP モジュールの読み込み

モジュールの読み込みには、**require()** 関数を使用します

HTTP モジュールは、require() 関数のパラメータに指定する事で読み込みます

```
var http = require('http');
```

HTTP モジュール

#### HTTP サーバーの作成

HTTP サーバーの作成には、HTTP モジュールの、**createServer()** 関数を使用します

```
var server = http.createServer();
```

#### ポートの指定

ポート番号は、**listen()** 関数のパラメータとして設定します

一般的な HTTP サーバーのポート番号は、80番

例では、ポート番号を変数で設定

```
server.listen(5XXXX);
```

ポート番号

## 3. Node.jsでWebアプリを作ってみよう

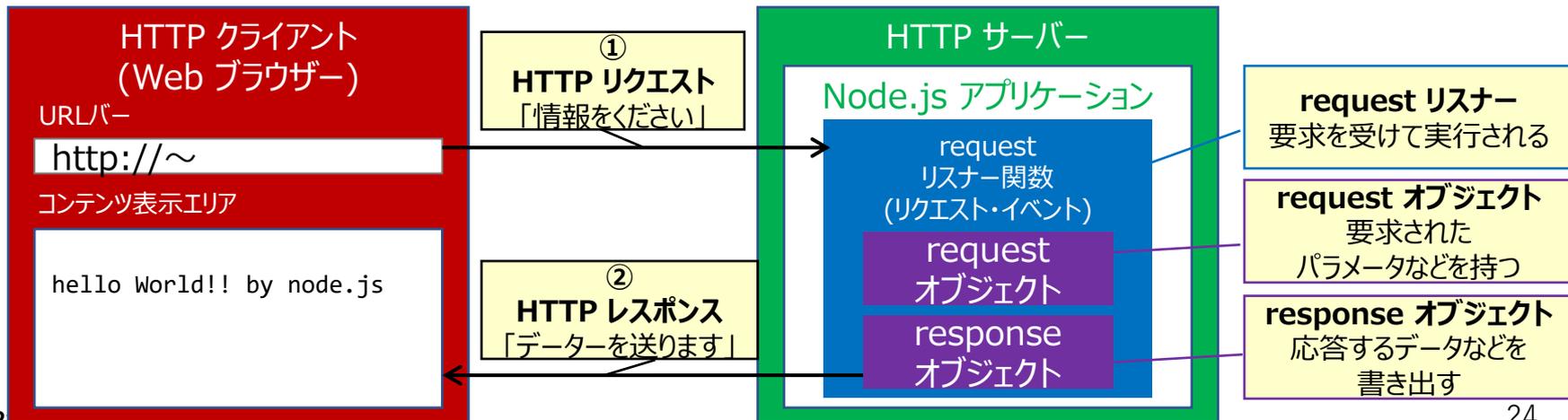
### 1) Hello.jsを作ってみよう-8 (プログラムの解説2)

#### HTTP サーバーの作成 (つづき)

##### 要求 (リクエスト) と、応答 (レスポンス)

HTTP サーバーと HTTP クライアントの関係は、HTTP リクエストと HTTP レスポンス で成り立ちます  
本書では、HTTP クライアントは Web ブラウザーを指します

- ① HTTP クライアントは HTTP サーバーへ要求をだします  
HTTP サーバーは、要求を受け取ると、request リスナー関数を実行します
- ② response オブジェクトを使用して、要求への応答データをクライアントへ送ります  
応答データの送信は、response オブジェクトを使用します



## 3. Node.jsでWebアプリを作ってみよう

### 1) Hello.jsを作ってみよう-9 (プログラムの解説3)

#### HTTP サーバーの作成 (つづき)

##### 応答 (レスポンス) の構成

HTTP レスポンス は、大きく分けて 2つで構成されます

##### レスポンス・ヘッダー

ステータスコード

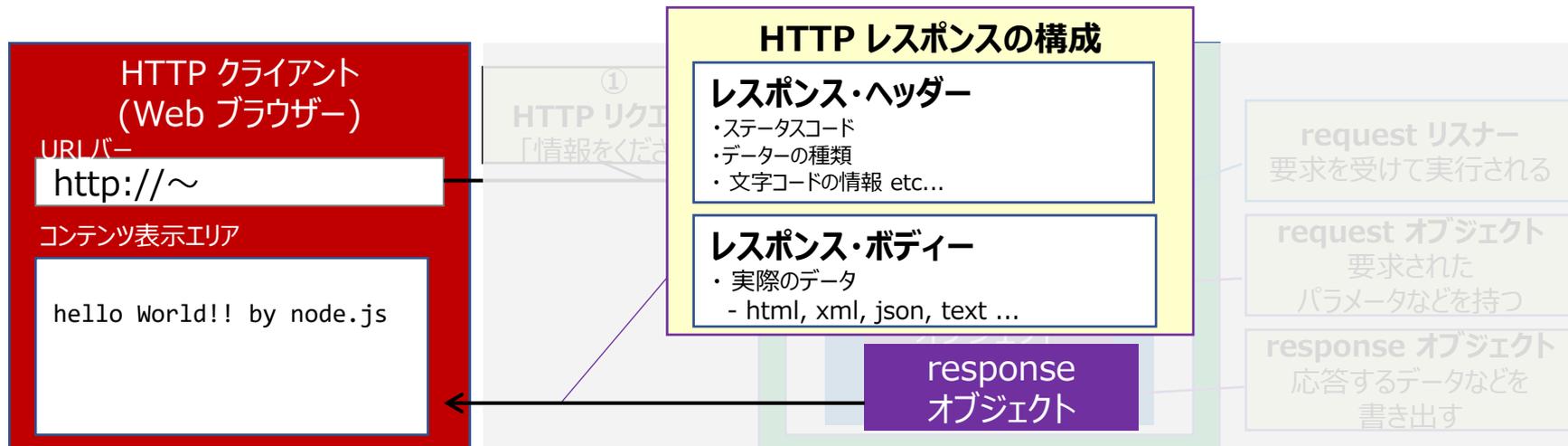
データの種類の設定 : HTML, XML, JSON, etc...

応答したデータの文字コード情報

etc...

##### レスポンス・ボディ

実際の応答データ : 様々な形式での応答が可能 (html, xml, json, text...)



## 3. Node.jsでWebアプリを作ってみよう

### 1) Hello.jsを作ってみよう-10 (プログラムの解説4)

#### HTTP サーバーの作成 (つづき)

##### リクエスト・リスナーの設定 (次ページも併せて参照)

先に作成した、createServer 関数のパラメーターに、**要求時に実行される関数**を設定します

要求時に実行される関数を **リクエスト・リスナー** と呼びます

リクエストリスナー は、名前のない関数 (無名関数) として設定

( JavaScript は、関数のパラメータとして、関数を指定する事ができます )

```
http.createServer(function(request, response){});
```

リクエストリスナーを無名関数で設定

request オブジェクト

response オブジェクト

リクエスト・リスナーのパラメーターは、以下の意味を持ちます

第1パラメーター：request - ブラウザーなどからの、HTTP 要求を取り扱うオブジェクト

第2パラメーター：response - ブラウザーなどへ戻す、HTTP 応答を取り扱うオブジェクト

## 3. Node.jsでWebアプリを作ってみよう

### 1) Hello.jsを作ってみよう-11 (プログラムの解説5)

#### Web アプリケーションの作成

※ Web アプリケーションとして動作する箇所の説明です

##### HTTP レスポンス・ヘッダー

HTTP レスポンス・ヘッダーは、応答データの属性を設定するために使用します

response オブジェクトの **writeHead()** 関数で設定します

response オブジェクトは、リクエスト・リスナーの第2パラメーターです

例では、ステータスコード(200 : 成功)、応答データのタイプ(text)、文字コード(utf-8) を設定している

```
response.writeHead(200,  
  {'Content-Type': 'text/plain; charset=utf-8'});
```

```
graph TD
    A[ステータスコード] --- B(200)
    C[コンテンツタイプ] --- D['Content-Type']
    E[文字コード] --- F[charset=utf-8]
```

コンテンツタイプ

文字コード

## 3. Node.jsでWebアプリを作ってみよう

### 1) Hello.jsを作ってみよう-12 (プログラムの解説6)

#### Web アプリケーションの作成 (つづき)

##### HTTP レスポンス・ボディ

HTTP クライアントへ応答する実際のデータを設定します

データの形式には、様々な形式があります

HTML、XML、JSON、テキスト、...

応答するデータの形式は、HTTP レスポンス・ヘッダーを使用して、HTTP クライアントに知らせる

response オブジェクトの **write()** 関数で設定します

例では、テキスト形式の 'hello world!! by Node.js' を設定している

```
res.write('Hello World!! by Node.js');
```

ブラウザへの応答

response オブジェクトの **end()** 関数で、HTTP レスポンスを終了します

```
res.end();
```

## 3. Node.jsでWebアプリを作ってみよう

### 1) Hello.jsを作ってみよう-13 (プログラムの解説7)

#### コンソール・ログ

簡易的なログ機能としてコンソールに出力する機能を持ちます

ログ機能は、**console.log()** 関数を使用します

簡単なデバッグなどに利用することができます

表示は、**PASE for i シェル (QP2TERM)** に表示されます

例では、+ 演算子 を使用し、文字リテラルと変数の連結をおこなっています

```
console.log('run server port = '+ ServerPort);
```



コンソールに表示したい文字

```
run server port = 59999
```

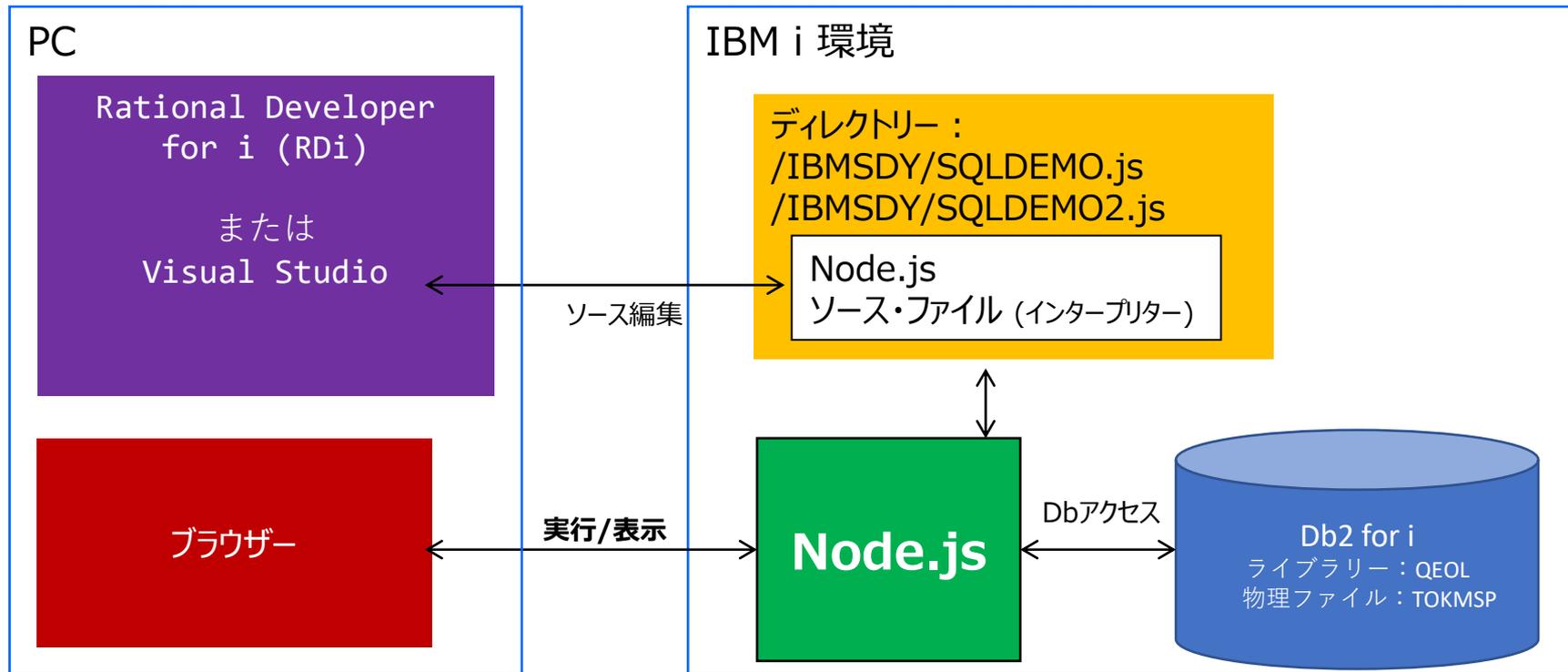
以上で、『プログラムの解説』は終了です。

### 3. Node.jsでWebアプリを作ってみよう

#### 2) 既存のデータベースにアクセスしてみよう-1 (実行環境イメージ)

Node.js のソース編集 を RDi で行います

RSE を使用して、IBM i の IFS のソースファイルを直接編集します.他の使い慣れたエディターでもOKです。



### 3. Node.jsでWebアプリを作ってみよう

#### 2) 既存のデータベースにアクセスしてみよう-2 (Db2ドライバーのダウンロード)

既存のライブラリーにあるデータベースにアクセスするプログラムを作成してみましょう。

最初に、Db2 for iデータベースにアクセスするためのドライバーをダウンロードします。

最新の「idb-connector」というモジュールをダウンロードします。

PASE環境にログインし「npm i idb-connector」  
コマンドを入力します。

```
> npm i idb-connector

> idb-connector@1.2.15 install /node_modules/idb-connector
> node-pre-gyp install --fallback-to-build

[idb-connector] Success: "/node_modules/idb-connector/lib/binding/Release/nap
i3-ibmi-ppc64/db2ia.node" is installed via remote
npm WARN saveError ENOENT: no such file or directory, open '/package.json'
npm notice created a lockfile as package-lock.json. You should commit this fi
le.
npm WARN enoent ENOENT: no such file or directory, open '/package.json'

=>
```



```
> pwd
/home/sawada
```

このまま数分待つと、自分のホームディレクトリー（pwdで確認）  
にダウンロードされます。（下記では、/home/Sawadaの下）

```
ディレクトリー... : /home/sawada/node_modules

オプションを入力して、実行キーを押してください。
2= 編集 3= コピー 4= 除去 5= 表示 7= 名前の変更
11= 現行ディレクトリーの変更 ...

OPT オブジェクト・リンク タイプ 属性 テキス
-- gauge DIR
-- glob DIR
-- has-unicode DIR
-- https-proxy-agent DIR
-- idb-connector DIR
-- inflight DIR
-- inherits DIR
-- is-fullwidth-code- > DIR
-- lru-cache DIR
```

idb-connectorの詳細については下記を参照してください。

<https://www.npmjs.com/package/idb-connector>

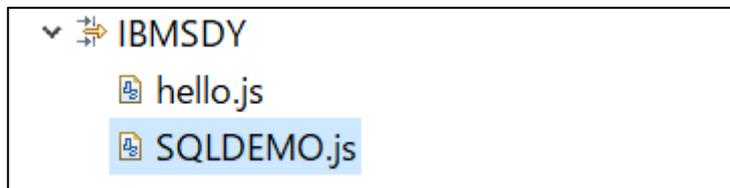
上記のように

`/home/sawada/node_modules/idb-connector`  
にダウンロードされました。これを使用します。

### 3. Node.jsでWebアプリを作ってみよう

#### 2) 既存のデータベースにアクセスしてみよう-3 (ソースコードの作成)

(1)と同様に、RDiで、IFS上の/IBMSDYフォルダーに、「SQLDEMO.js」を作成します。



下記のように入力し、保存してください。

```
var http = require('http');
var db2i = require('/home/sawada/node_modules/idb-connector');
var sql = 'select * from QEOL.TOKMSP';
var connection= new db2i.dbconn();
connection.conn('*LOCAL'); //Local dbに接続
var stm =new db2i.dbstmt(connection);
http.createServer(function (req, res) {

stm.exec(sql,function(rs){
res.writeHead(200, {'Content-Type': 'text/plain; charset=utf-8'});
res.write(JSON.stringify(rs)); //json形式で出力
statement.close();
connection.disconnect();
connection.close();
});
}).listen(59999);
console.log('server run:59999');
```

### 3. Node.jsでWebアプリを作ってみよう

#### 2) 既存のデータベースにアクセスしてみよう-4 (プログラムの実行)

PCでブラウザを起動し、以下のURLを実行します。  
xxxは、マシンのIPアドレス (又はホスト名)

http://xxx.xxx.xxx.xxx:59999/

: 成功すると、下記のようにブラウザに

json形式の物理ファイル：TOKMSPが表示されます。

(1)と同様に、PASE環境 (call qp2term) で、  
Node.jsのプログラムを実行します。

```
$
> PATH=/QOpenSys/pkgs/bin:$PATH
$
> export PATH
$
> node -v
v14.18.3
$
> node /IBMSDY/SQLDEMO.js
server run:59999
```



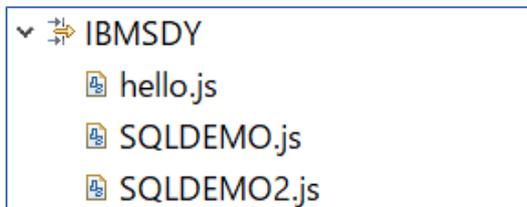
```
demo00:59999
[{"TKRANG": "01010", "TKNANK": "74 113", "TKNAKU": "阿井絵輝", "TKADR1": "東京都渋谷区", "TKADR2": "桜丘2-9", "TKTIKU": "02", "TKPOST": "150", "TKTELE": "03-604-9293", "TKZURI": "68500", "TKDURI": "4083300", "TKUZAN": "100000", "TKGENO": "1100000", "TKNYUM": "880427", "TKISME": "21", "TKISANG": "01020", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "阿井工業", "TKADR1": "東京都渋谷区", "TKADR2": "渋谷1-5", "TKTIKU": "02", "TKPOST": "150", "TKTELE": "03-635-956", "TKZURI": "42200", "TKDURI": "2719800", "TKZURI": "4483300", "TKUZAN": "670000", "TKGENO": "1150000", "TKNYUM": "880307", "TKISME": "21", "TKISANG": "01030", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "相川工業", "TKADR1": "東京都世田谷区", "TKADR2": "若林4-2-4", "TKTIKU": "06", "TKPOST": "154", "TKTELE": "03-984-8406", "TKZURI": "13620", "TKDURI": "243000", "TKZURI": "78600", "TKUZAN": "110000", "TKGENO": "1120000", "TKNYUM": "980619", "TKISME": "11", "TKISANG": "01040", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "東京朝日川", "TKADR1": "東京都目黒区", "TKADR2": "東吉田1-1-1", "TKTIKU": "01", "TKPOST": "140", "TKTELE": "03-847-7949", "TKZURI": "3028300", "TKDURI": "19083100", "TKZURI": "27670700", "TKUZAN": "9000000", "TKGENO": "1200000", "TKNYUM": "880402", "TKISME": "4", "TKISANG": "01050", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "阿井食品", "TKADR1": "東京都荒川区", "TKADR2": "荒川6-4-2", "TKTIKU": "17", "TKPOST": "116", "TKTELE": "03-380-6701", "TKZURI": "54100", "TKDURI": "2697800", "TKZURI": "4278600", "TKUZAN": "1300000", "TKGENO": "1350000", "TKNYUM": "880172", "TKISME": "11", "TKISANG": "01060", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "阿井自動車", "TKADR1": "東京都港区", "TKADR2": "芝公園1-2", "TKTIKU": "14", "TKPOST": "105", "TKTELE": "03-880-2332", "TKZURI": "54110", "TKDURI": "2819100", "TKZURI": "4539900", "TKUZAN": "1400000", "TKGENO": "9800000", "TKNYUM": "880525", "TKISME": "5", "TKISANG": "01070", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "相川", "TKADR1": "東京都新大塚", "TKADR2": "西9-1-6", "TKTIKU": "08", "TKPOST": "180", "TKTELE": "03-684-0319", "TKZURI": "38790", "TKDURI": "2011700", "TKZURI": "3914500", "TKUZAN": "960000", "TKGENO": "1000000", "TKNYUM": "880611", "TKISME": "10", "TKISANG": "01080", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "相川広業", "TKADR1": "東京都渋谷区", "TKADR2": "広尾3-9", "TKTIKU": "02", "TKPOST": "150", "TKTELE": "03-388-6386", "TKZURI": "51800", "TKDURI": "1491700", "TKZURI": "2256500", "TKUZAN": "290000", "TKGENO": "1408024", "TKISME": "11", "TKISANG": "01090", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "相川電機", "TKADR1": "東京都北区", "TKADR2": "滝の川1-7", "TKTIKU": "14", "TKPOST": "114", "TKTELE": "03-749-6271", "TKZURI": "877100", "TKDURI": "4791400", "TKZURI": "7136500", "TKUZAN": "940000", "TKGENO": "900000", "TKNYUM": "880519", "TKISME": "2", "TKISANG": "01100", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "相川建設", "TKADR1": "東京都港区", "TKADR2": "門前仲町", "TKTIKU": "08", "TKPOST": "105", "TKTELE": "03-622-1801", "TKZURI": "50190", "TKDURI": "2973300", "TKZURI": "4801100", "TKUZAN": "590000", "TKGENO": "1300000", "TKNYUM": "880615", "TKISME": "10", "TKISANG": "01110", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "相川設計事務所", "TKADR1": "東京都文京区", "TKADR2": "本郷1-2-6", "TKTIKU": "13", "TKPOST": "113", "TKTELE": "03-939-9124", "TKZURI": "446300", "TKDURI": "2664000", "TKZURI": "4530700", "TKUZAN": "740000", "TKGENO": "1600000", "TKNYUM": "880206", "TKISME": "6", "TKISANG": "01120", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "相川商事", "TKADR1": "東京都葛飾区", "TKADR2": "新小岩2-2-0", "TKTIKU": "21", "TKPOST": "124", "TKTELE": "03-866-7896", "TKZURI": "31600", "TKDURI": "1793800", "TKZURI": "3281400", "TKUZAN": "410000", "TKGENO": "3000000", "TKNYUM": "880306", "TKISME": "4", "TKISANG": "01130", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "東京工業", "TKADR1": "東京都中央区", "TKADR2": "日本橋4-2-3", "TKTIKU": "16", "TKPOST": "150", "TKTELE": "03-567-8588", "TKZURI": "63540", "TKDURI": "3409100", "TKZURI": "6789600", "TKUZAN": "680000", "TKGENO": "2500000", "TKNYUM": "880421", "TKISME": "5", "TKISANG": "01140", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "相川洋行", "TKADR1": "東京都練馬区", "TKADR2": "高野台2-1-4", "TKTIKU": "12", "TKPOST": "177", "TKTELE": "03-387-7983", "TKZURI": "19300", "TKDURI": "779300", "TKZURI": "1912000", "TKUZAN": "1200000", "TKGENO": "2400000", "TKNYUM": "880103", "TKISME": "11", "TKISANG": "01150", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "相川運輸", "TKADR1": "東京都杉並区", "TKADR2": "西武3-2-3", "TKTIKU": "04", "TKPOST": "187", "TKTELE": "03-691-8588", "TKZURI": "84200", "TKDURI": "133300", "TKZURI": "644800", "TKUZAN": "40000", "TKGENO": "2000000", "TKNYUM": "880621", "TKISME": "2", "TKISANG": "01160", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "東京建設", "TKADR1": "東京都目黒区", "TKADR2": "目黒4-2-3", "TKTIKU": "16", "TKPOST": "150", "TKTELE": "03-691-8588", "TKZURI": "14690", "TKDURI": "489300", "TKZURI": "1118300", "TKUZAN": "130000", "TKGENO": "1200000", "TKNYUM": "880315", "TKISME": "5", "TKISANG": "01170", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "相川会計事務所", "TKADR1": "東京都中央区", "TKADR2": "東銀座7-1-5", "TKTIKU": "08", "TKPOST": "103", "TKTELE": "03-895-5141", "TKZURI": "484100", "TKDURI": "2023300", "TKZURI": "3636000", "TKUZAN": "800000", "TKGENO": "1150000", "TKNYUM": "880222", "TKISME": "10", "TKISANG": "01180", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "相川商會", "TKADR1": "東京都横浜市", "TKADR2": "若葉2-2-2", "TKTIKU": "28", "TKPOST": "182", "TKTELE": "03-231-1579", "TKZURI": "67490", "TKDURI": "372000", "TKZURI": "607000", "TKUZAN": "910000", "TKGENO": "1400000", "TKNYUM": "880420", "TKISME": "3", "TKISANG": "01190", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "相川建設", "TKADR1": "東京都練馬区", "TKADR2": "練馬1-4-1", "TKTIKU": "23", "TKPOST": "155", "TKTELE": "03-761-9216", "TKZURI": "88000", "TKDURI": "5191100", "TKZURI": "6750100", "TKUZAN": "1020000", "TKGENO": "1000000", "TKNYUM": "880404", "TKISME": "6", "TKISANG": "01200", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "阿井自動車工業", "TKADR1": "東京都千代田区", "TKADR2": "神田神保町1-1", "TKTIKU": "09", "TKPOST": "101", "TKTELE": "03-384-3427", "TKZURI": "149100", "TKDURI": "410200", "TKZURI": "755000", "TKUZAN": "190000", "TKGENO": "1000000", "TKNYUM": "880302", "TKISME": "4", "TKISANG": "01210", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "東京建設", "TKADR1": "東京都杉並区", "TKADR2": "東上野3-2-4", "TKTIKU": "07", "TKPOST": "110", "TKTELE": "03-853-7744", "TKZURI": "1262500", "TKDURI": "7454100", "TKZURI": "11748000", "TKUZAN": "1950000", "TKGENO": "2500000", "TKNYUM": "880305", "TKISME": "2", "TKISANG": "01220", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "相川ビル", "TKADR1": "東京都港区", "TKADR2": "赤坂1-4-1", "TKTIKU": "08", "TKPOST": "104", "TKTELE": "03-731-1670", "TKZURI": "24280", "TKDURI": "1445400", "TKZURI": "2304800", "TKUZAN": "2800000", "TKGENO": "3000000", "TKNYUM": "880503", "TKISME": "11", "TKISANG": "01230", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "相川広業", "TKADR1": "東京都新宿区", "TKADR2": "西新宿4-3-1", "TKTIKU": "03", "TKPOST": "160", "TKTELE": "03-783-4171", "TKZURI": "49200", "TKDURI": "2732100", "TKZURI": "4232000", "TKUZAN": "670000", "TKGENO": "1400000", "TKNYUM": "880613", "TKISME": "3", "TKISANG": "01240", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "相川建築", "TKADR1": "東京都新宿区", "TKADR2": "大久保2-1-8", "TKTIKU": "03", "TKPOST": "180", "TKTELE": "03-888-2389", "TKZURI": "22700", "TKDURI": "1310500", "TKZURI": "2905100", "TKUZAN": "410000", "TKGENO": "1300000", "TKNYUM": "880615", "TKISME": "2", "TKISANG": "01250", "TKNANK": "74 113"}, {"TKRANG": "01010", "TKNANK": "アパテラ商事", "TKADR1": "東京都渋谷区", "TKADR2": "神宮前", "TKTIKU": "02", "TKPOST": "150", "TKTELE": "03-918-2851", "TKZURI": "70200", "TKDURI": "248900", "TKZURI": "676600", "TKUZAN": "120000", "TKGENO": "1800000", "TKNYUM": "880416", "TKISME": "10", "TKISANG": "01260", "TKNANK": "74 113"}]
```

### 3. Node.jsでWebアプリを作ってみよう

#### 2) 既存のデータベースにアクセスしてみよう-5 (ソースコードの入力)

下記のように入力し、保存してください。

同様に、RDiでIFS上のIBMSDYフォルダーに、SQLDEMO2.jsを作成します。



```

http = require('http');
db2i = require('/home/sawada/node_modules/idb-connector');
sql = 'select TKBANG,TKNAKN,TKNAKJ,TKADR1,TKADR2 from QEOL.TOKMSP';
connection= new db2i.dbconn();
ection.conn('*LOCAL'); //Local dbに接続
stm =new db2i.dbstmt(connection);
.createServer(function (req, res) {

exec(sql,function(rs){
writeHead(200, {'Content-Type': 'text/html; charset=utf-8'});
var html = '<table border="1">';
var data = '{ "file" : ' + JSON.stringify(rs) + '}';
data = eval("(" + data + ")");
html = html + '<tr>';
html = html + '<td>TKBANG</td><td>TKNAKN</td><td>TKNAKJ</td><td>TKADR1</td><td>TKADR2</td>';
html = html + '</tr>';
for(var i=0;i < Object.keys(data.file).length; i++){
html = html + '<tr>';
html = html + '<td>' + data.file[i].TKBANG + '</td>';
html = html + '<td>' + data.file[i].TKNAKN + '</td>';
html = html + '<td>' + data.file[i].TKNAKJ + '</td>';
html = html + '<td>' + data.file[i].TKADR1 + '</td>';
html = html + '<td>' + data.file[i].TKADR2 + '</td>';
html = html + '</tr>';
}
html = html + '</table>'
res.write(html);
stm.close();
connection.disconnect();
connection.close();

listen(59999);
console.log('server run:59999');

```

### 3. Node.jsでWebアプリを作ってみよう

#### 2) 既存のデータベースにアクセスしてみよう-5 (プログラムの実行)

PCでブラウザを起動し、以下のURLを実行します。

xxxは、マシンのIPアドレス (又はホスト名)

http://xxx.xxx.xxx.xxx:59999/

- 成功すると、下記のようにブラウザにHTMLでの表形式の物理ファイル：TOKMSPが表示されます。

(1)と同様に、PASE環境 (call qp2term) で、Node.jsのプログラムを実行します。

```
$
> export PATH=/QOpenSys/pkgsrc/bin:$PATH
$
> node /IBMSDY/SQLDEMO2.js
server run:59999
```



TKBANG	TKNAKN	TKNAKJ	TKADR1	TKADR2
01010	アイリョカ	阿井旅館	東京都渋谷区	桜ヶ丘2-9
01020	アイコウコウ	阿井工業	東京都渋谷区	渋谷1-3
01030	アイコウコウ	相川工業	東京都世田谷区	若林4-2-4
01040	アイヨコウヤ	阿井旅行社	東京都品川区	東五反田1-1-1
01050	アイヨクダクK.K	阿井食品K.K	東京都荒川区	荒川5-4-2
01060	アイシドウヤ	阿井自動車	東京都港区	芝公園1-2
01070	アイカメラ	相川カメラ	東京都新宿区	四谷1-1-6
01080	アイコウクK.K	相川広告K.K	東京都渋谷区	広尾3-9
01090	アイコウデンK.K	相川電機K.K	東京都北区	滝の川7-1-7
01100	アイカウキヤ	相川楽器店	東京都港区	虎ノ門3-2-1
01110	アイセキインムシヨ	相川設計事務所	東京都文京区	本郷1-2-5
01120	アイコウシヤ	相川商事	東京都葛飾区	新小岩2-2-0
01130	アイコウイヤ	愛工芸社	東京都港区	南麻布1-1-0
01140	アイカドクヤ	相川塗装店	東京都練馬区	高野台2-1-4
01150	アイカウクK.K	相川運輸K.K	東京都杉並区	西荻3-2-3
01160	アイカヒコウイ	相川病院	東京都目黒区	目黒4-2-3
01170	アイカウイヤムシヨ	相川会計事務所	東京都中央区	東銀座7-1-5
01180	アイカウイヤムシヨ	相川商会	東京都調布市	若葉2-2-2

### 3. Node.jsでWebアプリを作ってみよう

2) 既存のデータベースにアクセスしてみよう-6 (SQLDEMOのプログラム解説)

簡単にプログラムを解説します。

```
var http = require('http');
var db2i = require('/home/sawada/node_modules/idb-connector');
var sql = 'select * from QEOL.TOKMSP';
var connection= new db2i.dbconn();
connection.conn('*LOCAL'); //Local dbに接続
var stm =new db2i.dbstmt(connection);
http.createServer(function (req, res) {

stm.exec(sql,function(rs){
  res.writeHead(200, {'Content-Type': 'text/plain; charset=utf-8'});
  res.write(JSON.stringify(rs));
  stm.close();
  connection.disconnect();
  connection.close();
});
}).listen(59999);
console.log('server run:59999');
```

IBM iデータベースアクセスのためのモジュールを使うのに必要な設定

JSON形式を文字列化して、ブラウザに表示します

待ち受けのポート番号

### 3. Node.jsでWebアプリを作ってみよう

2) 既存のデータベースにアクセスしてみよう-7  
説明) 簡単にプログラムを解説します。

(SQLDEMO2プログラム解

```
var http = require('http');
var db2i = require('/home/sawada/node_modules/idb-connector');
var sql = 'select TKBANG,TKNAKN,TKNAKJ,TKADR1,TKADR2 from QEOL.TOKMSP';
var connection= new db2i.dbconn();
connection.conn('*LOCAL'); //Local dbに接続
var stm=new db2i.dbstmt(connection);
http.createServer(function (req, res) {
```

```
stm.exec(sql,function(rs){
res.writeHead(200, {'Content-Type': 'text/html; charset=utf-8'});
var html = '<table border="1">';
var data = '{ "file" : ' + JSON.stringify(rs) + '}';
data = eval ("(" + data + ")");
html = html + '<tr>';
html = html +
'<td>TKBANG</td><td>TKNAKN</td><td>TKNAKJ</td><td>TKADR1</td><td>TKADR2</td>';
html = html + '</tr>';
for(var i=0;i < Object.keys(data.file).length; i++){
html = html + '<tr>';
html = html + '<td>' + data.file[i].TKBANG + '</td>';
html = html + '<td>' + data.file[i].TKNAKN + '</td>';
html = html + '<td>' + data.file[i].TKNAKJ + '</td>';
html = html + '<td>' + data.file[i].TKADR1 + '</td>';
html = html + '<td>' + data.file[i].TKADR2 + '</td>';
html = html + '</tr>';
}
html = html + '</table>';
res.write(html);
stm.close();
connection.disconnect();
connection.close();
});
}).listen(59999);
console.log('server run:59999');
```

SOLDEMOプログラムのJSON  
形式をhtml形式に成形して  
います

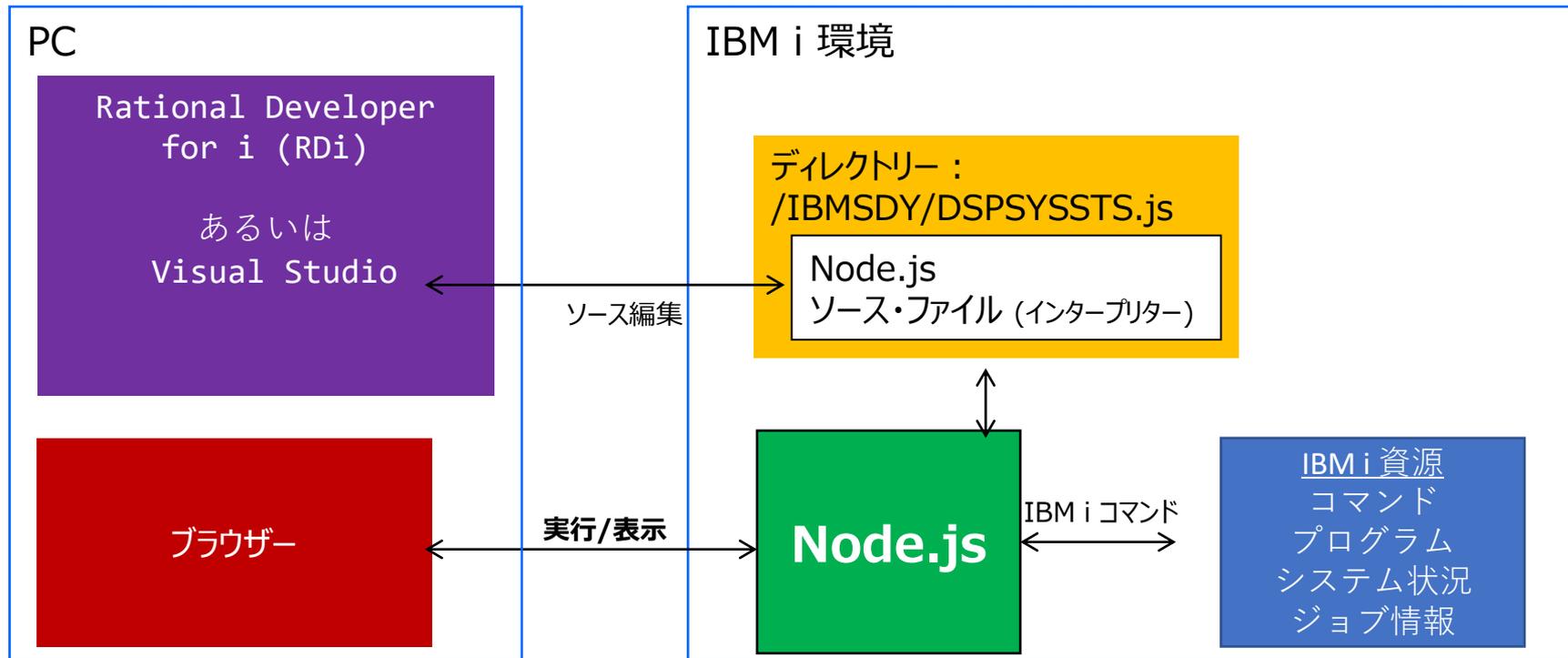
idb-connectorの詳細情報は→ <https://github.com/IBM/nodejs-idb-connector>

### 3. Node.jsでWebアプリを作ってみよう

#### 3) itoolkitを使ってIBM iのコマンドをWebでアクセスしてみよう-1 (実行環境イメージ)

Node.js のソース編集 を RDi で行います

RSE を使用して、IBM i の IFS のソースファイルを直接編集します.他の使い慣れたエディターでもOKです。



### 3. Node.jsでWebアプリを作ってみよう

#### 3) itoolkitを使ってIBM i コマンドをWebでアクセスしてみよう-2

次に既存のIBM iを利用するitoolkitを使って、Webでコマンドを起動してみましょう。

最初に、IBM iシステム資源にアクセスするモジュールをダウンロードします。

下記にあるようにitoolkitの最新をダウンロードしてください。

PASE環境にログインし「`npm install itoolkit`」  
コマンドを入力します。

```
> npm install itoolkit
> odbc@2.4.2 install /home/sawada/node_modules/odbc
> node-pre-gyp install --fallback-to-build

[odbc] Success: "/home/sawada/node_modules/odbc/lib/bindings/napi-v6/odbc.node" is installed via remote
npm WARN saveError ENOENT: no such file or directory, open '/home/sawada/package.json'
npm WARN enoent ENOENT: no such file or directory, open '/home/sawada/package.json'
```

このまま数分待つと、自分のホームディレクトリー  
(pwdで確認) にダウンロードされます。

```
ディレクトリー . . . : /home/sawada/node_modules

オプションを入力して、実行キーを押してください。
2= 編集 3= コピー 4= 除去 5= 表示 7= 名前
11= 現行ディレクトリーの変更 ...

OPT オブジェクト・リンク タイプ 属性
___ idb-pconnector DIR
___ inflight DIR
___ inherits DIR
___ is-fullwidth-code-point > DIR
___ itoolkit DIR
___ tru-cache DIR
___ make-dir DIR
___ minimatch DIR
___ minipass DIR
```

iToolkitの詳細については下記を参照してください。

<https://www.npmjs.com/package/itoolkit>

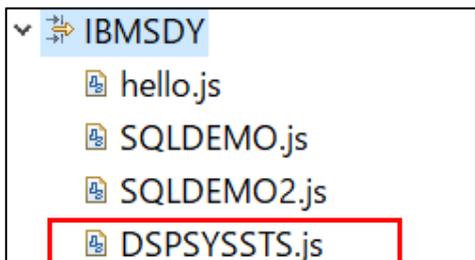
上記のように `/home/sawada/node_modules/itoolkit`  
にダウンロードされました。これを使います

## 3. Node.jsでWebアプリを作ってみよう

### 3) itoolkitを使ってIBM iコマンドをWebでアクセスしてみよう-3 (ソースコード入力)

下記のように入力し、保存してください。

(2)と同様に、RDiで、IFS上の/IBMSDYフォルダーに、DSPSYSSTS.jsを作成します。



```
var http = require('http');
var { Connection, CommandCall } = require('/home/sawada/node_modules/itoolkit');
var { parseString } = require('/home/sawada/node_modules/xml2js');
var connection = new Connection({
  transport: 'ssh',
  transportOptions: { host: 'IPアドレス', username: 'ユーザ-ID', password: 'パスワード' },
});
var command = new CommandCall({ type: 'sh', command: 'system -i DSPSYSSTS' });
connection.add(command);
http.createServer(function (req, res) {

connection.run((error, xmlOutput) => {
  if (error) {
    throw error;
  }
  parseString(xmlOutput, (parseError, result) => {
    if (parseError) {
      throw parseError;
    }
    res.writeHead(200, {'Content-Type': 'text/plain; charset=utf-8'});
    res.write(JSON.stringify(result));
  });
});
}).listen(59999);
console.log('server run:59999');
```



### 3. Node.jsでWebアプリを作ってみよう

#### 3) itoolkitを使ってIBM iコマンドをWebでアクセスしてみよう-5(プログラムの解説)

```
var http = require('http');
var { Connection, CommandCall } = require('/home/sawada/node_modules/itoolkit');
var { parseString } = require('/home/sawada/node_modules/xml2js');
var connection = new Connection({
  transport: 'ssh',
  transportOptions: { host: 'IPアドレス', username: 'ユーザーID', password: 'パスワード' },
});
var command = new CommandCall({ type: 'sh', command: 'system -i DSPSYSSTS' });
connection.add(command);

http.createServer(function (req, res) {
  connection.run((error, xmlOutput) => {
    if (error) {
      throw error;
    }
    parseString(xmlOutput, (parseError, result) => {
      if (parseError) {
        throw parseError;
      }
      res.writeHead(200, { 'Content-Type': 'text/plain; charset=utf-8' });
      res.write(JSON.stringify(result));
    });
  });
}).listen(59999);
console.log('server run:59999');
```

IBM i資源にアクセスするための  
itoolkitを使用するのに必要な設  
定です。

IBM iとの接続(IP,ユーザー  
IDなどを設定します

DSPSYSSTSのコマンドを  
定義しています。

JSON形式を文字列化して、  
ブラウザに表示します

## 4. 補足情報

(1) Node.jsの概要と特徴について

<https://www.i-cafe.info/column/serials/evo008>

(2) IBM i yum 導入ガイド

OSS協議会IBM iでのセミナー資料

[https://i5php.jp/wp-content/uploads/2019/09/yum\\_guide.pdf](https://i5php.jp/wp-content/uploads/2019/09/yum_guide.pdf)

(3) IBM i でNode-REDを動かす

OSS協議会IBM iでのセミナー資料

<https://i5php.jp/wp-content/uploads/2018/12/node-red.pdf>

(4) その他参考資料 The World of Node.js on IBM iセミナー資料 (英語です)

<https://www.scottklement.com/presentations/World%20of%20NodeJS%20on%20IBM%20i.pdf>

## 補足情報2

(5) IBM i OSSの概要の情報

<https://ibm.github.io/ibmi-oss-resources/>

(6) IBM i OSS Docsより、Node.js usage notes

<https://ibmi-oss-docs.readthedocs.io/en/latest/nodejs/README.html>

# OSS協議会IBM iのご紹介

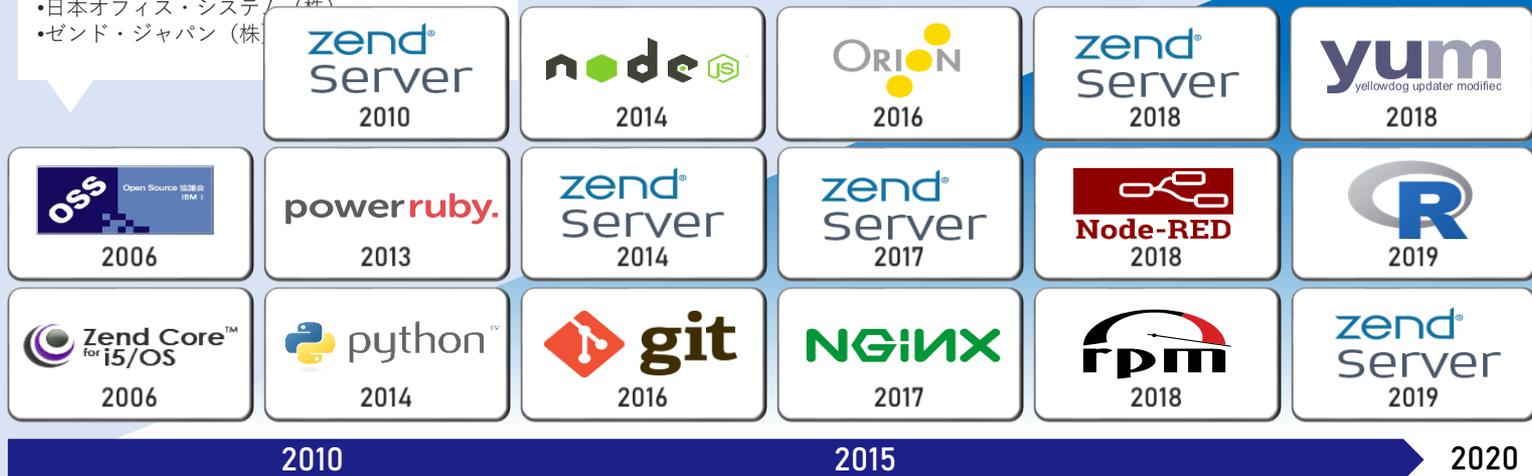
## OSS協議会-IBM i資料より抜粋

2006年、IBMビジネスパートナー  
5社が中心となって設立

- 日本ビジネスコンピューター (株)
- (株) 福島情報サービス
- トップランエムアンドアイ (株)
- 日本オフィス・システム (株)
- ゼンド・ジャパン (株)

5733-OPS (IBM i オープン・ソース・  
ソリューション) というライセン  
ス・プログラムとして提供開始

RPMとYumをサポートしたことで、こ  
れまで以上にIBM iにおけるオープ  
ンソースの導入や活用、管理が容易に。



詳細は、下記をご覧ください。多数のOSS関連の情報を掲載しています。

<https://i5php.jp/>

# IBM Power Salonのご案内



IBM Powerユーザーのための自由な語り場がオープンします。  
オンラインサロンで、お客様同士、IBMスペシャリストと繋がりませんか  
是非、お気軽にご参加ください。

日時：第6回 2022年 4月13日(水) 9:00-10:00

内容：お客様によるDX事例、クラウド活用事例  
IBM i/AIX/Linuxの技術情報、サポート情報  
IBMスペシャリストによるQ&A 他



毎月第2水曜日	主なトークテーマ (予定)
2022年 2月 9日	IBM i
2022年 3月 9日	IBM i
2022年 4月13日	IBM Power セキュリティと可用性
2022年 5月11日	IBM i
2022年 6月 8日	Power10

主催：日本アイ・ビー・エム（株）IBM Power 事業部

参加方法：オンライン開催  
どなたでも参加可能、無料  
事前申し込み不要

ご参加URL：<https://ibm.biz/Powersalon-webex>

お問い合わせ：[NO1POWER@jp.ibm.com](mailto:NO1POWER@jp.ibm.com)

## IBM Community Japan

## IBM i Club ご案内

## ■ IBM i Clubとは

- 自社システムとしてIBM iをご利用いただいている皆様同士で、各社の工夫や事例を紹介し合ったり、ディスカッションをしていただく場です。
- IBMの技術者も参加し、IBM iの最新情報のご提供や、ディスカッションに入らせていただくこともあります。
- 年5回(予定)、それぞれテーマを変えて開催します。

## ■ 2021年のテーマ例 (ご参加の皆様からいただいた課題をもとにテーマを選出しました)

- IBM i ユーザーハイブリッドクラウドへの道
- IBM i 人材育成の勘所
- 新技術の活用
- IBM i ユーザーのDX
- IBM i の優位性と今後

## ■ 開催期間

- 2022年2月24日から2022年12月末(予定)

## ■ 開催内容

- オンライン(Webex)セッションの開催: 2022年2月24日(木)を第1回とし年間5回(予定)
- 情報共有・ディスカッションの場: コミュニケーションツール(Slack)をご利用いただき、セッション以外の時も情報交換など可能です。

## ■ ご参加にあたってのお願い・ご注意点

- 守秘義務をお守りください。
- 営業活動を目的としたご参加はお断りいたします。
- IBM i Clubお申し込みには、事前にIBM Community Japanのメンバー登録が必要です。
- 開催期間の途中からのご参加も可能です。



## ■ 2022年開催予定

\*日時・内容が変更になる可能性があります

	日時	実施内容	
1	2/24 (終了)	最新情報	「数字で見るIBM i小辞典」IBM 佐々木
		事例紹介	「利用部門からの要望」にどう対応しているか? トクラス(株)様
		ディスカッション	テーマ: これからのIT部門の役割
2	5月	最新情報	「IBM i 新リリース発表」
		事例紹介	(調整中)
		ディスカッション	テーマ: 人材確保・人材育成
3	7月	最新情報	
		事例紹介	
		ディスカッション	テーマ: 新技術の活用(仮)
4	9月	最新情報	
		事例紹介	
		ディスカッション	テーマ: DX(仮)
5	11月	最新情報	
		事例紹介	
		ディスカッション	

## ■ コース詳細・お申込み

<https://www.ibm.com/ibm/jp/ja/ibmcommunityjapan.html>

## ■ ご参考 昨年 (2021年) 開催内容

[https://higherlogicdownload.s3.amazonaws.com/IMWUC/2fde9da6-6e7d-43b4-bae3-7f25168bbbd0/UploadedImages/japan/2022/2021\\_IBM\\_i\\_Club.pdf](https://higherlogicdownload.s3.amazonaws.com/IMWUC/2fde9da6-6e7d-43b4-bae3-7f25168bbbd0/UploadedImages/japan/2022/2021_IBM_i_Club.pdf)

# IBM i 情報

IBM i ポータル・サイト  
<https://ibm.biz/ibmijapan>

月イチIBM Power情報セミナー「IBM Power Salon」  
<https://ibm.biz/power-salon>

IBM i World 2021 オンデマンド・セミナー  
<https://ibm.biz/iworld2021>

IBM i ホワイトペーパー 2021年日本語版  
<https://www.ibm.com/downloads/cas/JB8AXO9V>

IBM i Club (日本のIBM i ユーザー様のコミュニティー)  
<https://ibm.biz/ibmiclubjapan>

i Magazine (IBM i 専門誌。春夏秋冬の年4回発刊)  
<https://www.imagazine.co.jp/>

IBM i 情報 Facebook  
<https://www.facebook.com/iusersjapan>

IBM i 研修サービス (i-ラーニング社提供)  
<https://www.i-learning.jp/service/it/iseriess.html>

Fix Central (HW・SWのFix情報提供)  
<https://www.ibm.com/support/fixcentral/>

IBM My Notifications (IBM IDの登録 [無償] が必要)  
「IBM i」「9009-41G」などPTF情報の必要な製品を  
選択して登録できます。  
<https://www.ibm.com/support/mynotifications>

IBM i 7.4 技術資料  
<https://www.ibm.com/docs/ja/i/7.4>

IBM i 各バージョンのライフサイクル  
<https://www.ibm.com/support/pages/release-life-cycle>

IBM i 以外のSWのライフサイクル (個別検索)  
<https://www.ibm.com/support/pages/lifecycle/>



ワークショップ、セッション、および資料は、IBMによって準備され、IBM独自の見解を反映したものです。それらは情報提供の目的のみで提供されており、いかなる読者に対しても法律的またはその他の指導や助言を意図したのではなく、またそのような結果を生むものでもありません。本資料に含まれている情報については、完全性と正確性を期するよう努力しましたが、「現状のまま」提供され、明示または暗示にかかわらずいかなる保証も伴わないものとします。本資料またはその他の資料の使用によって、あるいはその他の関連によって、いかなる損害が生じた場合も、IBMは責任を負わないものとします。本資料に含まれている内容は、IBMまたはそのサプライヤーやライセンス交付者からいかなる保証または表明を引き出すことを意図したもので、IBMソフトウェアの使用を規定する適用ライセンス契約の条項を変更することを意図したものでなく、またそのような結果を生むものでもありません。

本資料でIBM製品、プログラム、またはサービスに言及していても、IBMが営業活動を行っているすべての国でそれらが使用可能であることを暗示するものではありません。本資料で言及している製品リリース日付や製品機能は、市場機会またはその他の要因に基づいてIBM独自の決定権をもっていつでも変更できるものとし、いかなる方法においても将来の製品または機能が使用可能になると確約することを意図したものではありません。本資料に含まれている内容は、読者が開始する活動によって特定の販売、売上高の向上、またはその他の結果が生じると述べる、または暗示することを意図したもので、またそのような結果を生むものでもありません。パフォーマンスは、管理された環境において標準的なIBMベンチマークを使用した測定と予測に基づいています。ユーザーが経験する実際のスループットやパフォーマンスは、ユーザーのジョブ・ストリームにおけるマルチプログラミングの量、入出力構成、ストレージ構成、および処理されるワークロードなどの考慮事項を含む、数多くの要因に応じて変化します。したがって、個々のユーザーがここで述べられているものと同様の結果を得られると確約するものではありません。

記述されているすべてのお客様事例は、それらのお客様がどのようにIBM製品を使用したか、またそれらのお客様が達成した結果の実例として示されたものです。実際の環境コストおよびパフォーマンス特性は、お客様ごとに異なる場合があります。

IBM、IBM ロゴ、ibm.com、Db2、Rational、Power、POWER8、POWER9、AIXは、世界の多くの国で登録されたInternational Business Machines Corporationの商標です。

他の製品名およびサービス名等は、それぞれIBMまたは各社の商標である場合があります。

現時点での IBM の商標リストについては、[www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml) をご覧ください。

インテル、Intel、Intel ロゴ、Intel Inside、Intel Inside ロゴ、Centrino、Intel Centrino ロゴ、Celeron、Xeon、Intel SpeedStep、Itanium、およびPentium は Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。

Linuxは、Linus Torvaldsの米国およびその他の国における登録商標です。

Microsoft、Windows、Windows NT および Windows ロゴは Microsoft Corporationの米国およびその他の国における商標です。

ITILはAXELOS Limitedの登録商標です。

UNIXはThe Open Groupの米国およびその他の国における登録商標です。

JavaおよびすべてのJava関連の商標およびロゴは Oracleやその関連会社の米国およびその他の国における商標または登録商標です。