Developing with Node.js on iMX Developer's Kit

Copyright 2017 © Embedded Artists AB

Develop with Node.js on iMX Developer's Kits



Embedded Artists AB

Davidshallsgatan 16 SE-211 45 Malmö Sweden

http://www.EmbeddedArtists.com

Copyright 2017 © Embedded Artists AB. All rights reserved.

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written permission of Embedded Artists AB.

Disclaimer

Embedded Artists AB makes no representation or warranties with respect to the contents hereof and specifically disclaim any implied warranties or merchantability or fitness for any particular purpose. Information in this publication is subject to change without notice and does not represent a commitment on the part of Embedded Artists AB.

Feedback

We appreciate any feedback you may have for improvements on this document. Send your comments by using the contact form: <u>www.embeddedartists.com/contact</u>.

Trademarks

All brand and product names mentioned herein are trademarks, services marks, registered trademarks, or registered service marks of their respective owners and should be treated as such.

Table of Contents

1	Document Revision History 4
2	Introduction 5
2.1	Conventions5
3	Getting started 6
3.1	Add Node.js to Yocto image 6
3.2	Hello world
3.3	Simple web server7
4	WebStorm9
4.1	Install Node.js9
4.2	Install WebStorm
4.3	ECMAScript 6 13
4.4	Local debugging14
4.5	Remote deployment 14
16	
4.0	Run application on target 18
4.0 5	Run application on target 18 Troubleshooting

1 Document Revision History

Revision	Date	Description
А	2017-01-16	First release

2 Introduction

Node.js is a Javascript runtime environment that has become quite popular when developing different kinds of applications, for example, web server applications.

This document shows you how to install Node.js on the target file system and how to get started with your development. If you need to learn how to develop with Node.js or Javascript there are many resources online. A few of these are listed below.

https://nodejs.org/en/docs/

https://www.tutorialspoint.com/nodejs/index.htm

https://www.tutorialspoint.com/javascript/

Additional documentation you might need:

- The Getting Started document for the board you are using.
- The Working with Yocto document

2.1 Conventions

A number of conventions have been used throughout to help the reader better understand the content of the document.

Constant width text - is used for file system paths and command, utility and tool names.

```
$ This field illustrates user input in a terminal running on the
development workstation, i.e., on the workstation where you edit,
configure and build Linux
```

This field illustrates user input on the target hardware, i.e., input given to the terminal attached to the COM Board

This field is used to illustrate example code or excerpt from a document.

This field is used to highlight important information

3 Getting started

The instructions in this document have been tested on a virtual machine running **lubuntu 16.04**. The document "Working with Yocto to build Linux" has a chapter that explains how to create a VMware based virtual machine running lubuntu.

If you are an experienced Linux user it shouldn't be a problem using another Linux distribution with the instructions below as a guideline.

3.1 Add Node.js to Yocto image

The default Yocto images provided by Embedded Artists don't contain Node.js support. This can be added by modifying the local.conf file in your build. See the document "Working with Yocto to build Linux" for more details about building images. These instructions will add Node.js version **6.9.2**.

1. Open local.conf. Replace <build dir> with your build directory.

```
$ nano <build dir>/conf/local.conf
```

2. Find the IMAGE_INSTALL_append variable and add the lines below. The package called "nodejs-npm" installs a package manager for Node.js. Using this package manager you can install additional packages/modules. The third line installs build tools (compiler, linker) on the target file system. This can be needed if a Node.js module must be built from source during installation (when using npm). The last line isn't really related to Node.js. This line installs a SFTP server which can be useful when doing remote deployment.

```
nodejs \
nodejs-npm \
packagegroup-core-buildessential \
openssh-sftp-server \
```

 There are two more lines to add to local.conf. The first selects which version of Node.js to use. The second line was needed to solve "openssl" related build issues.

```
PREFERRED_VERSION_nodejs ?= "6.9.2"
PACKAGECONFIG append pn-nodejs = " openssl"
```

- 4. Save the file and exit the editor: CTRL+X followed by Y and Enter.
- Now build your image. In this example we are using a "core-image-base" build, but replace this with the image you are building.

```
$ bitbake core-image-nase
```

When the image has been built don't forget to deploy the image on the target. For more information see the "Working with Yocto" document.

NOTE: Node.js layers are only included in the **4.1.15** branch and were added to ea-yocto-base January 11, 2017. If you are using an older branch or revision you need to update.

3.2 Hello world

It is now time to create the first application and verify that Node.js is working. Please note that you must have deployed the new image on the target, booted into Linux, and having a console/terminal

application connected to the target. The "Getting Started" document contains instructions of how to use a console/terminal application.

1. Create an application file

```
# nano hello.js
```

Add the line below to the file. This line prints "Hello world".

```
console.log('Hello world')
```

- Save the file and exit: CTRL+X followed by Y and Enter.
- 4. Start the application

```
# node hello.js
Hello world
```

3.3 Simple web server

It is common to use Node.js when developing web applications. This example shows how a really simple web server can be created.

1. First get the IP address of the target since this is needed in a later step. In the example below the IP address is 192.168.1.130.

2. Create the application file.

```
# nano web.js
```

Add the code below to the file. This code is originally from <u>https://nodejs.org/dist/latest-v6.x/docs/api/synopsis.html</u>. Please note that you may need to change the IP address (hostname variable) to the IP address retrieved in step 1.

```
const http = require('http');
const hostname = '192.168.1.130';
const port = 3000;
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});
server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

- 4. Save the file and exit: CTRL+X followed by Y and Enter.
- 5. Start the application.

```
# node web.js
Server running at http://192.168.1.130:3000/
```

6. Start a web browser and enter the address shown in the console. You should see the message "Hello World" in the web browser.

4 WebStorm

There are many different editors and development environments for Node.js. For minor applications a basic text editor, such as nano or vi (on Linux), can be used. For more complex applications a more complete development environment, supporting for example syntax highlighting, code completion, and debugging, is often preferred. In this chapter we are going to describe how to install and use WebStorm from JetBrains.

4.1 Install Node.js

Before installing WebStorm it is recommended to install Node.js on your development computer. A lot of the development can be done on the computer and then deployed to the target board. As previously mentioned lubuntu 16.04 is used as development computer when writing these instructions.

You can use the package manager to install Node.js. This will give you Node.js version 4.2.6.

```
$ sudo apt-get install nodejs
```

Since the target file system will have version 6.9.2 it is however recommended to use the same version on the development computer. It is possible to download and install this version directly from the Node.js website.

On this link you can find different versions: https://nodejs.org/en/download/releases/.

The instructions below download and unpacks version 6.9.2 for a 64-bit Linux computer.

```
$ wget https://nodejs.org/download/release/v6.9.2/node-v6.9.2-
linux-x64.tar.gz
$ tar -xzvf node-v6.9.2-linux-x64.tar.gz
```

4.2 Install WebStorm

Please note that WebStorm is a commercial tool, but it can be used for 30 days for free.

- Go to <u>https://www.jetbrains.com/webstorm/</u> and click the "Download" button. A tar.gz file will then be downloaded (when writing these instructions the file was called WebStorm-2016.3.2.tar.gz).
- 2. Unpack the file. This will create a new directory (for these instructions the directory was called WebStorm-163.9166.30)
- 3. The directory contains a file called Install-Linux-tar.txt that describes how to install/start WebStorm. Basically what you need to do is run the webstorm.sh script.

\$./webstorm.sh

 When WebStorm is started you will be asked to activate the license. In this case we are evaluating WebStorm and choose "Evaluate for free" as shown in Figure 1. You must also accept the license agreement.

WebStorm License Activa	tion – + ×
○ Activate	Buy WebStorm
Evaluation is free for 30 days.	
Tell me about new product features as th	ey come out:
Email address (optional)	
Ev	aluate Exit

Figure 1 - WebStorm License activation

<u>ws</u>	License Agreement for WebStorm 2016.3.2 - + ×
i	To start your evaluation you must accept license agreement located at https://www.jetbrains.com/store/license.html
	Accept Cancel

Figure 2 - Accept agreement

5. You will then be asked for the initial configuration. We used the default settings as shown in Figure 3.

WebStorm Initial Configuration - + ×					
Keymap scheme:	Default for XWin				
IDE theme:	IntelliJ 🔹				
Editor colors and fonts:	Default 🔻				
▶ Click to preview					
You can use File Settings to cor	figure any of these settings later.				
Enable opening files and projects from the command line					
Script path: /usr/local/bin/webstorm					
Create desktop entry (integrate in system menu)					
For all users					
	ок Skip				

Figure 3 - WebStorm initial configuration

 Click on "Create New Project" as shown in Figure 4 to create a new project. Select the type "Empty Project" and specify a location as shown in Figure 5. When the project has been created it will look like Figure 6.

ws	Welcome to WebStorm - + ×
	WS
	WebStorm Version 2016.3.2
	🜟 Create New Project
	🧁 Open
	Check out from Version Control +
	🕸 Configure 🗸 🛛 Get Help 🗸

Figure 4 - Create a WebStorm project

WS	New Project - + ×
Empty Project	New project
😈 HTML5 Boilerplate	
🚺 Web Starter Kit	Location: /home/user/WebstormProjects/hello
🏶 React App	
🗹 Twitter Bootstrap	
🖪 Foundation	
🕼 AngularJS	
🔇 Angular CLI	
🏶 React Native	
🕼 Node.js Express App	
🏽 PhoneGap/Cordova App	
🗐 Yeoman	
淤 Meteor App	
🔦 Dart	
	Create
4	

Figure 5 - Empty WebStorm project



Figure 6 - WebStorm project

7. We need to specify which version of Node.js to use when running the application locally. Go to File → Settings in the menu and then select "Languages & Frameworks" → "Node.js and NPM". In the "Node interpreter" field specify the path to Node.js we installed in section 4.1 above. Figure 7 shows how the "Settings" dialog can look like when the interpreter has been chosen. Before closing the window click the "Enable" button in the "Code Assistance" field.

ws			Settings	5			- + ×
0	6	\supset	Languages & Fram	eworks > Node.j	s and NPM	For current pro	oject Reset
	Appearance & Behavior		Node interpreter:	~/node-v6.9.2-lir	nux-x64/bin/n	ode	6.9.2 🔻 …
	Keymap		Coding Assistance				
►	Editor		Nada is Casali				
	Plugins		Node.js Core u	brary is not enabl	ed. Enable		
►	Version Control		Packages		\sim		
	Directories		Package	Ver	sion	Latest	+
►	Build, Execution, Deployment		npm	3.10.9		➡ 4.0.5	_
▼	Languages & Frameworks						+
	JavaScript	G					
	Schemas and DTDs	ē					
	Dart	G					
	Node.js and NPM	G					
	Stylesheets	ē					
	Template Data Languages	Ē					
	TypeScript	Ē					
				0	K Canc	el Apply	Help

Figure 7 - Node.js settings dialog

- Click the OK button to close the "Settings" dialog.
- It is now time to create the application file. Go to File → New and click "JavaScript File" in the menu. Enter a name of the file. In this example we call it "hello".



Figure 8 - New Javascript file

10. The file will only contain a header when created. Add the line shown in Figure 9. This is the same application as show in section 3.2 above.



Figure 9 - Hello world application

11. Right-click on "hello.js" in the Project view and select "Run hello.js" to start the application. The output will be shown in the console window as illustrated in the bottom of Figure 9.

4.3 ECMAScript 6

If you try to run the application described in 3.3 you will get some errors as shown in Figure 10. The reason is that the code contains JavaScript constructs that was introduced in the JavaScript version called ECMAScript 6. By default WebStorm is using ECMAScript 5.1.



Figure 10 - Simple web server with errors

To change the version, go to File \rightarrow Settings in the menu. Then select "Languages & Frameworks" \rightarrow JavaScript and change the version to ECMAScript 6 as shown in Figure 11.

ws	Settings - +	×
٩	Languages & Frameworks > JavaScript B For current project	
Plugins	JavaScript language version ECMAScript 5.1 🔻	
Version Control	Prefer <u>S</u> trict mode ECMAScript 3 ECMAScript 5.1	
Directories	Only type-based completic JavaScript 1.8.5	
Build, Execution, Deployment	ECMAScript 6	
Languages & Frameworks	React JSX	
► JavaScript 🖷	1000	
▶ Schemas and DTDs 👘		
Dart		
Node.js and NPM 💿		
► Stylesheets		
Template Data Languages 🛛 🖻		
► TypeScript		
XSLT		
XSLT File Associations		
▶ Tools		
	OK Cancel Apply Help	

Figure 11 - Change JavasScript language version

4.4 Local debugging

Debugging on the development computer is quite simple. All you need to do is set a breakpoint in the code by clicking on the row. Then right-click on "hello.js" in the project view and select "Debug hello.js". Figure 12 shows a debug session where the debugger has stopped on a breakpoint.

WS	hello - [~/WebstormProjects/hello]/hello.js - WebStorm 2016.3.2	- + ×
<u>File Edit View Navigate Code Refacto</u>	r R <u>u</u> n <u>T</u> ools VC <u>S</u> <u>W</u> indow <u>H</u> elp	
🖿 hello 👌 💼 hello.js 👌		💽 hello.js 🔻 🕨 🕺 🔍
🗊 Project 👻 😌 💠 📂	💼 hello.js ×	
hello ~/WebstormProjects/hello	Enable File Watcher to transpile ECMAScript 6 to ECMAScript 5 using Babel?	Yes No
neuo.js	1 /**	*
	2 * Created by user on 2017-01-12.	
	<pre>4 const http = require('http');</pre>	
	5 const bostname = '127 0 0 1';	
	7 const port = 3000;	
	8 enset server - http://realizesarver//realizes) -> {	
	10 res.statusCode = 200;	
	<pre>11 res.setHeader('Content-Type', 'text/plain'); 12 res.setHeader('Kelle Heald): 13 res.setHeader('Kelle Heald): 14 res.setHeader('Kelle Heald): 15 res.setHeader('Kelle Heald): 16 res.setHeader('Kelle Heald): 17 res.setHeader('Kelle Heald): 17 res.setHeader('Kelle Heald): 18 res.setHeader('Kelle Header('Kelle Heade</pre>	
	12 <u>res</u> .ello(necco wor cu(ii); 13 });	
	14	
	<pre>15 server.tisten(port, nostname, () => { 16 console.log(`Server running at http://\${hostname}:\${port}/`);</pre>	
	17 });	
Debug 🙉 hello.js		☆ - <u>⊥</u>
C Debugger E Console → Scripts →	王 ≥ ≥ 조 ≒ ■	
Frames →* ■ V	ariables	→*
□ ↓ ▼ ↓ ▼	Local	
hello.js:9	exports = Object	
C Module. compile(), module.js:5	Image:	
👝 🔄 Module. extensionsjs(), modu 💶	http = Object	
Module.load(), module.js:487	module = Module	
" Is tryModuleLoad(). module.is:446 "	III port = 3000	
		9:20 LF≑ UTF-8 ≑ 🖶 📿

Figure 12 - Debug session in WebStorm

4.5 Remote deployment

It is possible to deploy the application from within WebStorm, that is, upload it to the target.

Go to File \rightarrow Settings and then "Build, Execution, Deployment" \rightarrow Deployment and click on the plus icon as shown in Figure 13.

ws		Settings	- + ×
0	Keymap	(Insert) = recution, Deployment > Deployment	nt 🐵 For current project
►	Editor		
	Plugins		
►	Version Control		
	Directories		
v	Build, Execution, Deployment	Not configured	
	Debugger		
	▼ Deployment @	Please add a web	server to configure
	Options		
	Coverage 🐵		
	Docker		
	Docker Registry		
	Languages & Frameworks		
	► JavaScript @		
	Schemas and DTDs		
		ОК	Cancel Apply Help

Figure 13 - Create a deployment

Give the connection a name and then choose "SFTP" as server type as shown in Figure 14. Click "OK".

ws	Add Server - + >	<
?	Name:	
_	iMX Target ↑↓	
	<u>Т</u> уре:	
	💀 SFTP 🔽	
	Project files are deployed to a remote host via SFTP	
	OK Cancel	

Figure 14 - Add server

In the "Connection" settings window specify the IP address of the target in the "SFTP host" field. Set the user name (root) and password (pass). When this has been done click on the "Test SFTP connection" button to verify that the connection is working. If it is working you can click in the browse button (three dots) by the "Root path" field and choose where to upload the files. In this case we have chosen the home directory of the user "root". All of this is shown in Figure 15.

NOTE: By default the user "root" is not permitted to use an SSH connection. See section 5.1 how to permit the user "root" to login.

Build, Execu	tion, Deployment > I	Deployment 🐵 For current proje	ct Reset			
+ — 🛅 🐺	Na <u>m</u> e: iMX Targe	t				
	Connection Mappings Excluded Paths					
	Visible only for t	his project				
	<u>T</u> ype: 📴 SFTP	*				
	Project files a	re deployed to a remote host via	SFTP			
	Upload/download pro	ject files				
	SFTP h <u>o</u> st:	192.168.1.130	Test SFTP connection			
	<u>P</u> ort:	22				
	<u>R</u> oot path:	/home/root	Autodetect			
	<u>U</u> ser name:	root	Login as anonymous			
	Auth type:	Password				
	Pa <u>s</u> sword:	••••	Save password			
	Advanced opt	ions				
	Browse files on server	,				
	<u>W</u> eb server root	URL: http://192.168.1.130	Open			
		OK Canc	el Apply Help			

Figure 15 - Deployment connection settings

Go to the "Mappings" tab and select the "Deployment path" as shown in Figure 16.

	Settings	- + ×
Build, Execut	tion, Deployment > Deployment 🐵 For current project	Reset
👼 🗂 — + iMX Targ	Na <u>m</u> e: iMX Target	
	Connection Mappings Excluded Paths	
	Use this server as default	
	Local path:	
	/home/user/WebstormProjects/hello	
	D <u>e</u> ployment path on server 'iMX Target':	
	1	
	Web path on server 'iMX Target':	
	/	
	Project URL: http://192.168.1.130/	
	Add another mapping	

Figure 16 - Deployment mappings

To deploy the application right-click on the project and then go to Deployment \rightarrow "Upload to iMX Target" as shown in Figure 17.

🗊 Project	• ⊕ ≑ ∳ • +	븕 hello.js 🗴	
🔪 🖿 hello ~/V	New		her to transpile ECMAScript 6 to ECMAScrip
🛻 hello.j: 🎼 External I	₩ Cu <u>t</u>	Ctrl+X	d by user on 2017-01-13.
	Copy Path	Ctrl+Shift+C	<pre>p = require('http');</pre>
	Copy as Plain Text Copy Relative Path	Ctrl+Alt+Shift+C Ctrl+V	<pre>tname = '192.168.1.130'; t = 3000; ver = http.createServer((reg, res) => {</pre>
	Find <u>U</u> sages Find in <u>P</u> ath Repl <u>a</u> ce in Path	Alt+F7 Ctrl+Shift+F Ctrl+Shift+R	<pre>tatusCode = 200; stHeader('Content-Type', 'text/plain'); nd('Hello World\n'); rten(nest_besterme_() => {</pre>
	<u>I</u> nspect Code <u>R</u> efactor	,	<pre>le.log(`Server running at http://\${hostname</pre>
	Add to F <u>a</u> vorites Show Image Thumbnails	Ctrl+Shift+T	
	Local <u>H</u> istory Synchronize 'hello'	,	
	Show in File Manager PC Directory <u>P</u> ath	ManFM Ctrl+Alt+Shift+2	
	🛍 Compare With	Ctrl+D	
🔲 Upload sele	Mark Directory as Remove BOM	,	
	🕹 Deployment	•	👍 <u>U</u> pload to iMX Target
	Create Gist		▲ <u>D</u> ownload from iMX Target ② Sync with Deployed to iMX Target

Figure 17 - Deploy application

It is also possible to automatically deploy the application, for example, each time you save the project. Go to File \rightarrow Settings and then "Build, Execution, Deployment" \rightarrow Deployment \rightarrow Options. As shown in Figure 18 you can select to automatically upload the files.

WS	Sett	ings	- + ×
٩	Build, Execution, Deplo	yment > Deployment > Options	For current project Reset
Appearance & Behavior	Exclude items by name:	.svn;.cvs;.idea;.DS_Store;.git;.hg) 1r
 Editor Plugins Version Control Directories Build, Execution, Deployment Debugger Denloyment 	Operations logging: Stop operation on the Overwrite up-to-date Preserve files timesta Delete target items w Create empty directo	Brief e first error files amps vhen source ones do not exist (who ries	hen transferring from Project view or R
Options	Prompt when overwr	iting or deleting local items	
Coverage Docker Docker Registry Languages & Frameworks Tools	Upload changed files aut Upload external chan Override default perr Override default perr Warn when uploading ov	omatically to the default sever iges missions on files: missions on folders: er <u>n</u> ewer file:	On explicit save action (Ctrl+S) (none) (none) No V
	 SFTP Advanced Options (IDE level setting)	

Figure 18 - Deployment options

4.6 Run application on target

Besides deploying an application to the target it is also possible to start the application on the target from within WebStorm. Go to Run \rightarrow "Edit Configurations" in the menu as shown in Figure 19.

ello - [~/WebstormProjects/hello] - .../hello.js - WebS

•	Ru	<u>n T</u> ools VC <u>S W</u> indow <u>H</u> elp	
	▶	R <u>u</u> n 'hello.js'	Shift+F10
1	<u>ж</u> с	<u>D</u> ebug 'hello.js'	Shift+F9
-	188	Run 'hello.js' with Co <u>v</u> erage	
E	▶	Run	Alt+Shift+F10
1	<u>ж</u>	Debug	Alt+Shift+F9
3	₽	Edit Configu <u>r</u> ations	
5	Ð	Import Test Results	•
6		Stop	Ctrl+F2
8	P	Show Running List	
9 16	+	Step <u>O</u> ver	F8
11	*	Force Step O <u>v</u> er	Alt+Shift+F8
12 13	<u>M</u>	Step Into	F7
14	<u>"</u>	Force Step I <u>n</u> to	Alt+Shift+F7
15	2	Smart Ste <u>p</u> Into	Shift+F7
17	<u>×</u>	Step Ou <u>t</u>	Shift+F8
	M	Run to <u>C</u> ursor	Alt+F9
	M	Force Run to Cur <u>s</u> or	Ctrl+Alt+9
	П	<u>P</u> ause Program	
1	₽	Resume Program	F9
		Evaluate Expression	Alt+F8
		Quick Evaluate Expression	Ctrl+Alt+8
	Þ	Show Execution Point	Alt+F10
		Toggle <u>L</u> ine Breakpoint	Ctrl+F8
		Toggle Temporary Line Breakpoint	Ctrl+Alt+Shift+F8
de		Toggle <u>B</u> reakpoint Enabled	-
	8:	View Brea <u>k</u> points	Ctrl+Shift+F8

Figure 19 - Edit "Run configurations"

Click on the "Browse" button beside the "Node interpreter" field as shown in Figure 20.

Run/Debug Configurations –		
<u>N</u> ame: hello.js	🗌 Share 🗹 Single insta	nce only
Configuration Browser / Live Edit V8 Profiling		
Node interpreter:	~/node-v6.9.2-linux-x64/bin/node (Project) 6.9.2	• MT
Node <u>p</u> arameters:		
Working directory:	/home/user/WebstormProjects/hello	
JavaScript <u>f</u> ile:	hello.js	
Application parameters:		1
Environment variables:		
 ■ Before launch: Activate too + - // ↑ ↓ 	l window There are no tasks to run before launch	

Figure 20 - New node interpreter

Click on the "plus" icon and select "Add Remote" as shown in Figure 21.

<u>ws</u>	Node.js Interpreters	- + ×	
~/node-v6.9.2-linux-	x64/bin/node (Project)	6 9.2 +	
		A	dd Local
		A	dd Remote
			\smile
<u>N</u> ode interpreter:	~/node-v6.9.2-linux-x64/bin/node		
Np <u>m</u> package:	de-v6.9.2-linux-x64/lib/node_modules/npm 3.10	.9 🔻	
	OK Cancel	Help	

Figure 21 - Add remote interpreter

Specify the IP address as well as user name (root) and password (pass). Finish by clicking on OK.

ws	Configure Node.js Remote Interpreter		- +	×
○ <u>V</u> agrant ○ <u>D</u> eploym	ent configuration 🧿 SSH Credentials 🔘 Docker			
<u>H</u> ost:	192.168.1.130	Port:	22	
<u>U</u> ser name:	root			
Auth type:	Password 🔹			
P <u>a</u> ssword:	••••• Sav <u>e</u> password			
Node.js interpreter path:	/usr/bin/node			
	ок	(ance	ī

Figure 22 - Remote interpreter settings

You can now run the application on target by right-clicking on the target and select "Run hello.js". Please note that you must have deployed the application before you can run it.

5 Troubleshooting

5.1 Allow user "root" to use an SSH connection

By default the user "root" is not permitted to login via an SSH connection. By following these instructions "root" will be permitted to login through an SSH connection. It is, however, not recommended to use on a final application, but during development it can be permitted.

1. Open the configuration file for the SSH server

nano /etc/ssh/sshd_config

2. Find the line that starts with #PermitRootLogin and remove the '#' (hash) character. If you cannot find this line just add it to the file (without the hash)

PermitRootLogin yes

- 3. Save the file and exit the editor (in nano it is Ctrl-X followed by Y and Enter).
- 4. Restart the SSH server

/etc/init.d/sshd restart