Version: 4.x

Debugging

This guide will help you get started with debugging. There are multiple things you can debug and you need a different configuration for each one:

1. **Debug your client code** - No configuration needed, just open devtools on your browser.
2. **Debug your server code** - `yoshi start --debug`
3. **Debug your tests** - `yoshi test --debug`

When debugging your server/tests you'll need to configure the debugger, depend on your preferred way to debug.

#Enable Inspector

When started with the `--debug` option, Yoshi will allow to attach NodeJS debugger to the relevant child process with the default host and port. You can configure the default port by: `--debug=XXXX`

#Break before the code starts

When started with the `--debug-brk` option, Yoshi will allow to attach NodeJS debugger and the relevant child process won't start until debugger will be attached. You can configure the default port by: `--debug-brk=XXXX`.

#Inspector Clients

Several commercial and open source tools can connect to Node's Inspector and there for can debug Yoshi tasks. Basic info on these follows:

#Chrome DevTools 55+

- **Option 1**: Open `chrome://inspect` in a Chromium-based browser. Click the Configure button and ensure your target host and port are listed.
- **Option 2** - ✅ **Recommended**: Install the Chrome Extension NIM (Node Inspector Manager):https://chrome.google.com/webstore/detail/nim-node-inspector-manage/gnhhdgbaldcilmgcpfddgdbkhjohddkj

#Visual Studio Code 1.10+

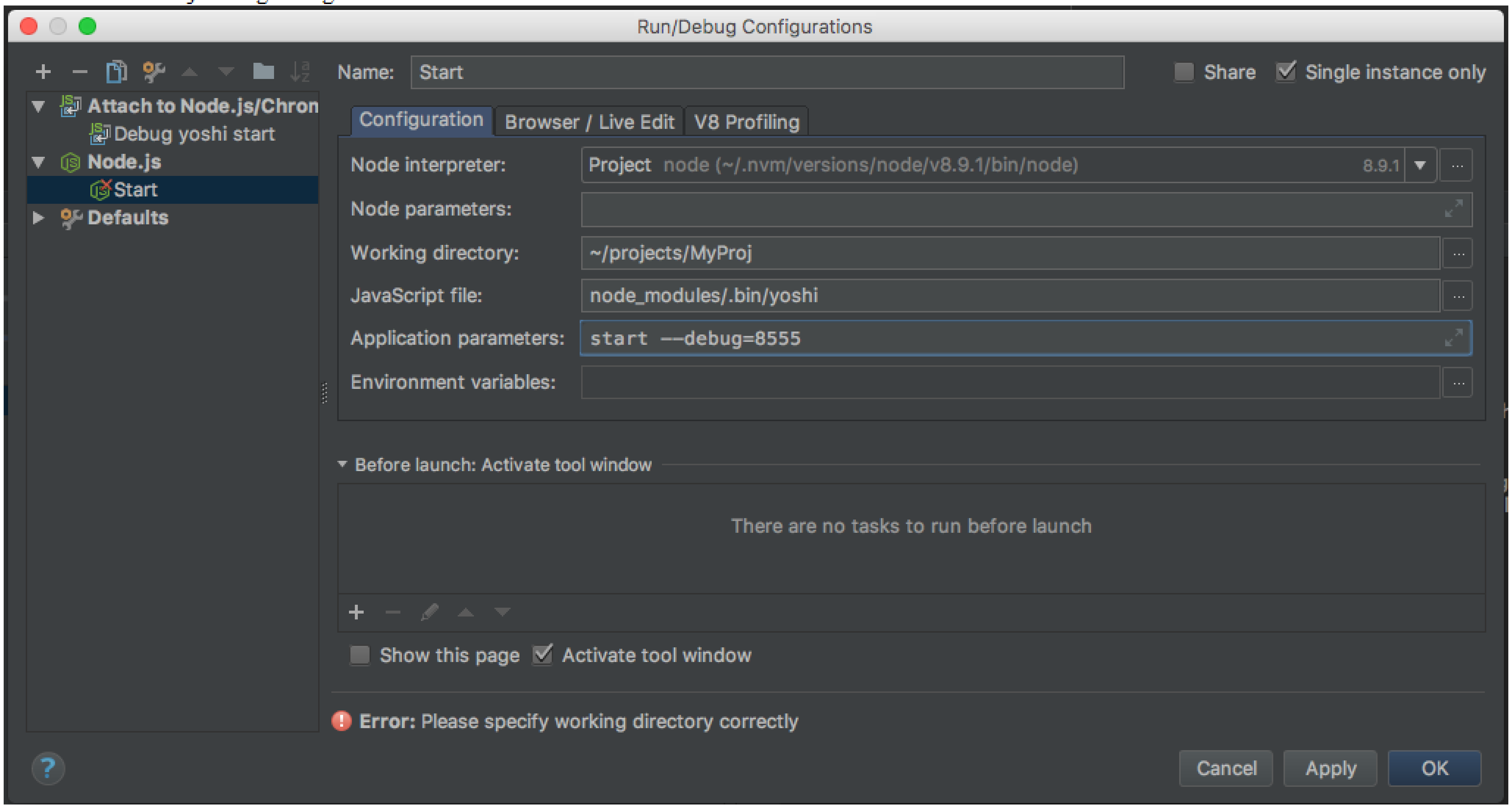
- In the Debug panel, click the settings icon to open `.vscode/launch.json`. Select "Node.js" for initial setup.
- 📌 You must tell vscode the target debugging port, otherwise vscode will try to debug Yoshi's main process in random generated port, so add `"port" : 9229` (or the port you choose)
- Example launch.json -

Copy

```
{
"name": "Run Tests",
"type": "node",
"request": "launch",
"args": ["test", "--debug-brk"],
"port": 9229,
"program": "${workspaceFolder}/node_modules/.bin/yoshi"
}
```

#JetBrains WebStorm 2017.1+ and other JetBrains IDEs

- Create a new Node.js debug configuration



Run/Debug Configurations

- Attach to Node.js/Chron
  - Debug yoshi start
- Node.js
  - Start
- Defaults

Name: Start — Share [✓ Single instance only]

Configuration | Browser / Live Edit | V8 Profiling

Node interpreter: Project node (~/.nvm/versions/node/v8.9.1/bin/node) 8.9.1
Node parameters:
Working directory: ~/projects/MyProj
JavaScript file: node_modules/.bin/yoshi
Application parameters: start --debug=8555
Environment variables:

Before launch: Activate tool window

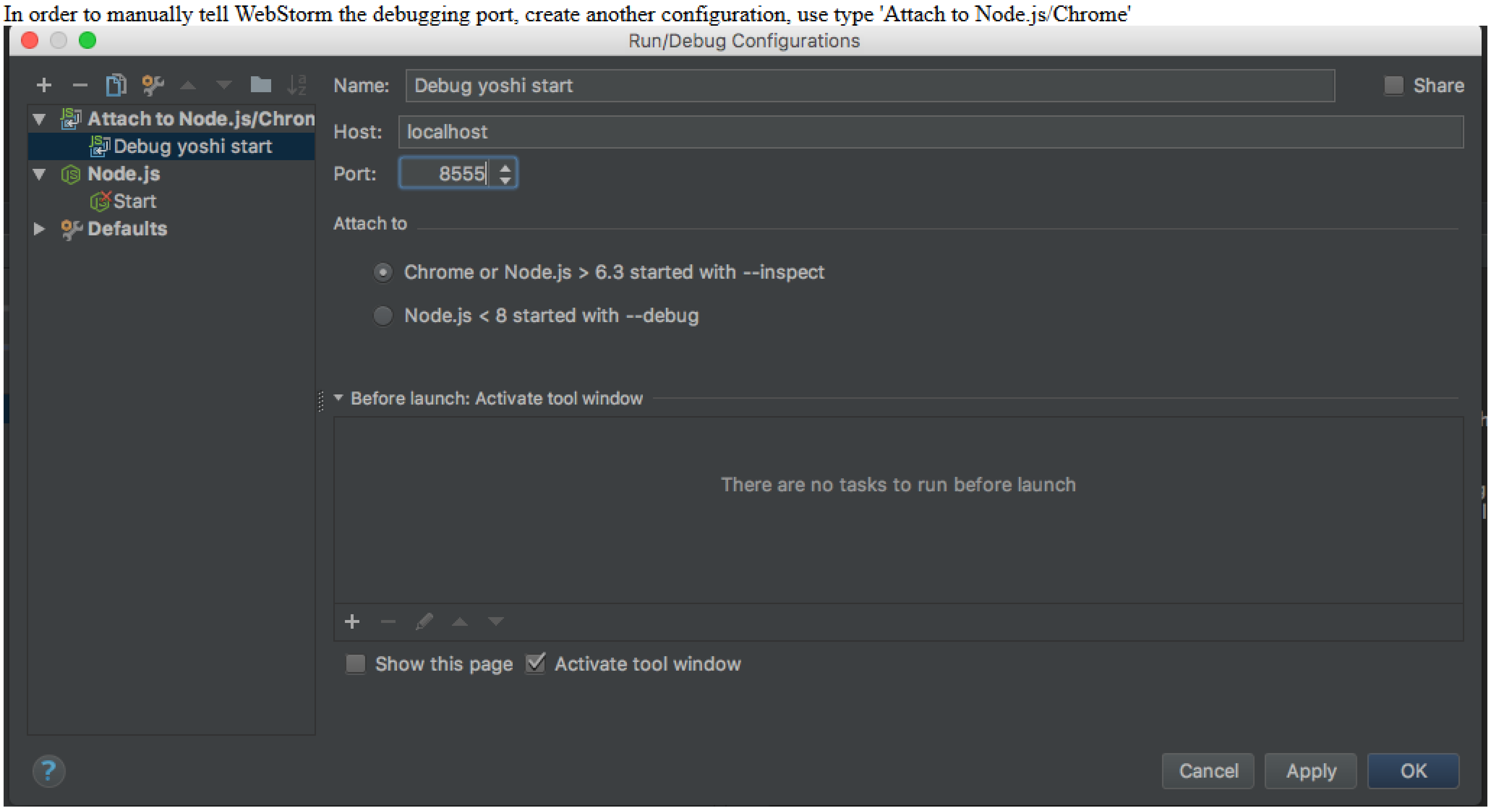
There are no tasks to run before launch

Show this page [✓ Activate tool window]

Error: Please specify working directory correctly

Cancel Apply OK

- In order to manually tell WebStorm the debugging port, create another configuration, use type 'Attach to Node.js/Chrome'



Run/Debug Configurations

- Attach to Node.js/Chron
  - Debug yoshi start
- Node.js
  - Start
- Defaults

Name: Debug yoshi start — Share
Host: localhost
Port: 8555

Attach to
[✓ Chrome or Node.js > 6.3 started with --inspect]
Node.js < 8 started with --debug

Before launch: Activate tool window

There are no tasks to run before launch

Show this page [✓ Activate tool window]

Cancel Apply OK

- Press debug in order to start the remote debugger configuration then start (without debugging) the 'Node.js' configuration

#Debugging Puppeteer E2E tests

#Debugging in watch mode

- Run `yoshi test --watch` to run in watch mode
- Press `d` in the watch menu to activate debug mode

#Debugging by default configuration

- Set `devtools: true` in `jest-yoshi.config`

#Adding breakpoints in the browser

- In order to add a breakpoint to browser, add `await debugBrowser();` to your test.

Edit this page
Last updated on 5/17/2020 by ***Amit Dahan***