



# Node.JS

Node.js is an open source, cross-platform, JavaScript runtime built on Chrome's V8 JavaScript engine, for developing server and client side applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.



*Node's goal is to provide an easy way  
to build scalable network programs*

Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#). Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, [npm](#), is the largest ecosystem of open source libraries in the world.

Important [security releases](#), please update now!

## Download for macOS (x64)

**v6.11.3 LTS**

Recommended For Most Users

**v8.4.0 Current**

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

Or have a look at the [LTS schedule](#).

Sign up for [Node.js Everywhere](#), the official Node.js Weekly Newsletter.

 **LINUX FOUNDATION** [COLLABORATIVE PROJECTS](#)

[Report Node.js issue](#) | [Report website issue](#) | [Get Help](#)

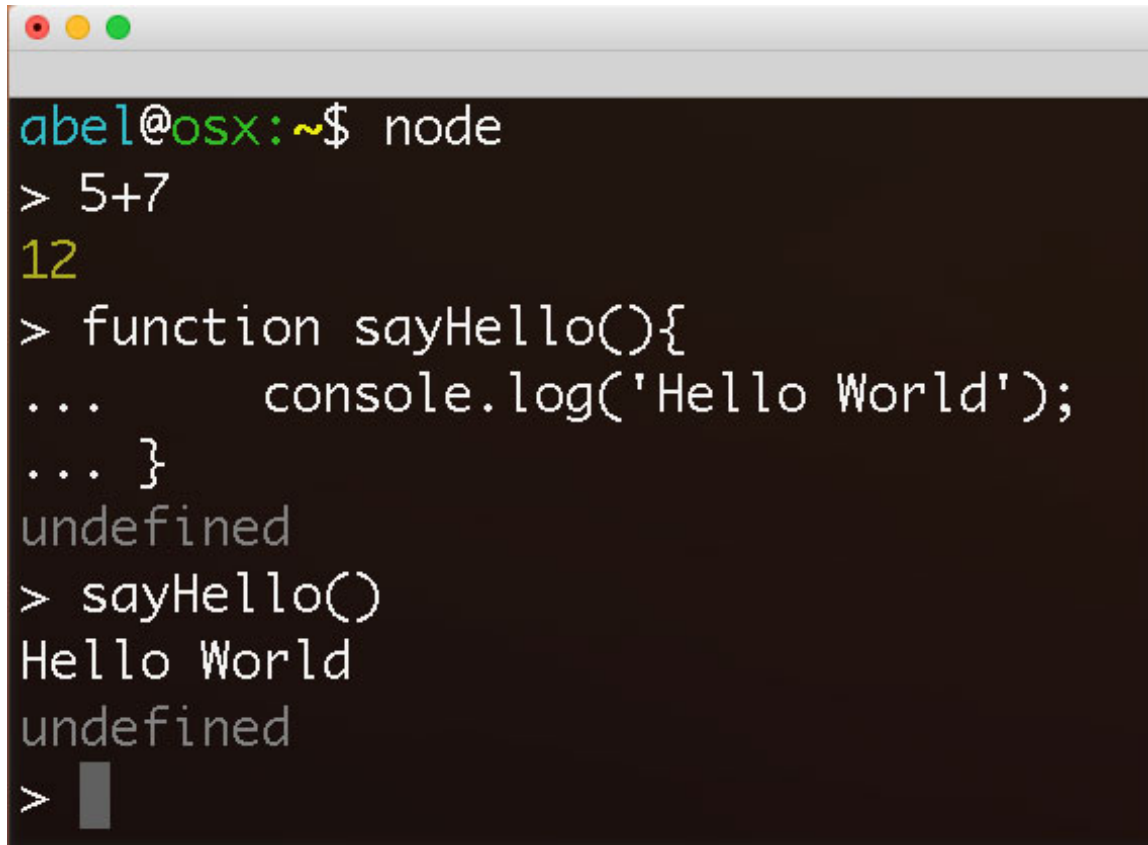
© 2017 Node.js Foundation. All Rights Reserved. Portions of this site originally © 2017 Joyent.

Node.js is a trademark of Joyent, Inc. and is used with its permission. Please review the [Trademark Guidelines](#) of the Node.js Foundation.

Linux Foundation is a registered trademark of The Linux Foundation.

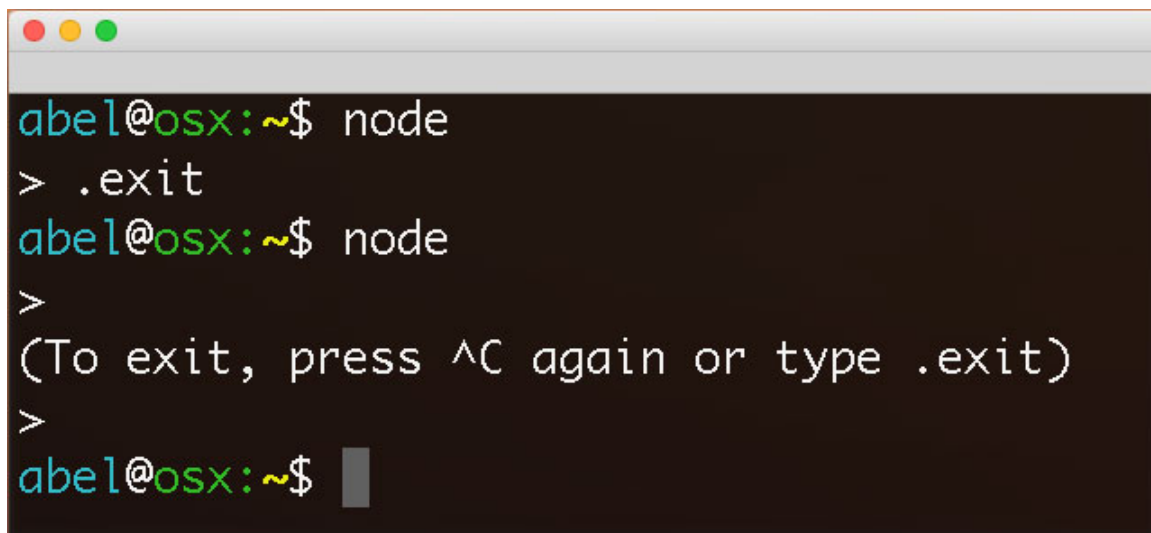
Linux is a registered trademark of Linus Torvalds.

# Console Use

A terminal window with a dark background and a light gray title bar. The title bar has three colored window control buttons (red, yellow, green) on the left. The terminal shows a Node.js REPL session. The prompt is 'abel@osx:~\$'. The user enters 'node'. The prompt changes to '>'. The user enters '5+7'. The output is '12'. The user enters a function definition: 'function sayHello(){ ... console.log('Hello World'); ... }'. The output is 'undefined'. The user enters 'sayHello()'. The output is 'Hello World'. The user enters 'undefined'. The output is 'undefined'. The prompt is '>' with a cursor.

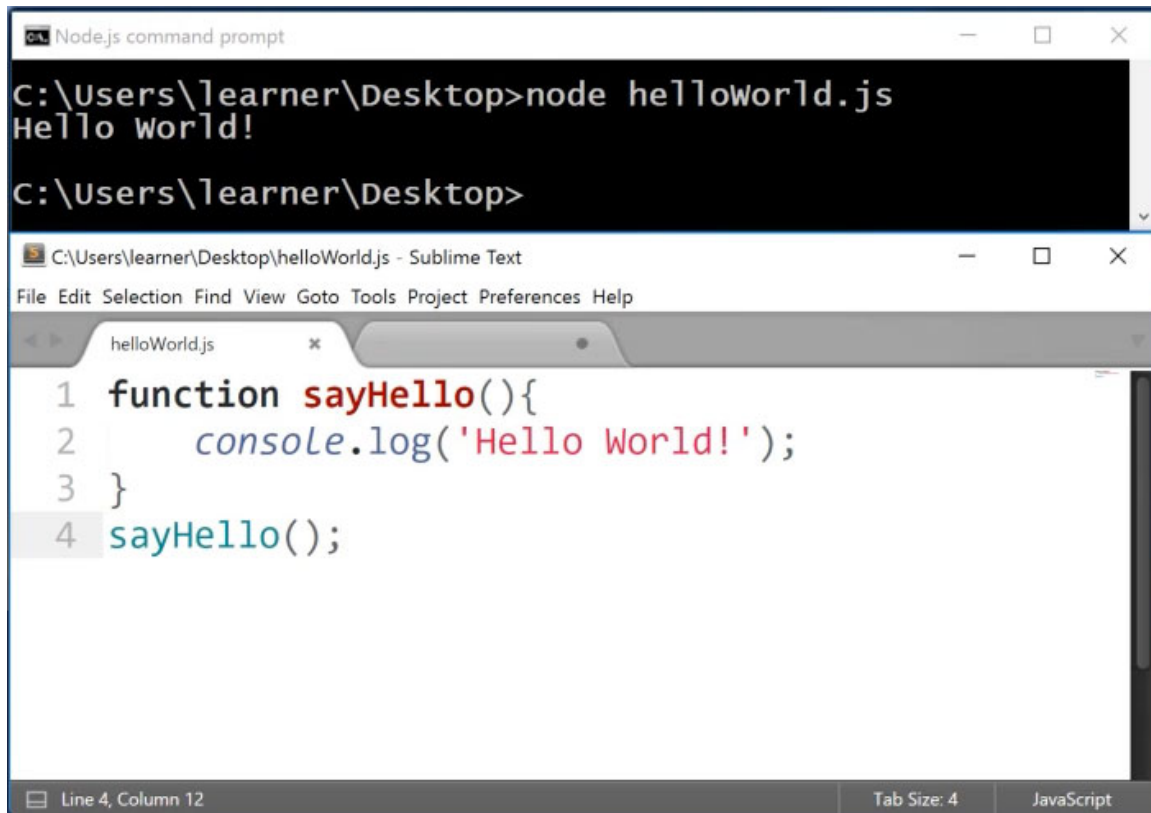
```
abel@osx:~$ node
> 5+7
12
> function sayHello(){
...     console.log('Hello World');
... }
undefined
> sayHello()
Hello World
undefined
>
```

# Exit Console

A screenshot of a macOS terminal window with a dark background. The window has a title bar with three colored buttons (red, yellow, green) on the left. The terminal shows a Node.js REPL session. The prompt is 'abel@osx:~\$' in cyan and green. The user enters 'node' in white. The prompt changes to '>' in white. The user enters '.exit' in white. The prompt returns to 'abel@osx:~\$' in cyan and green. The user enters 'node' in white. The prompt changes to '>' in white. A message '(To exit, press ^C again or type .exit)' is displayed in white. The prompt changes back to '>' in white. The user enters a blank line, and the prompt returns to 'abel@osx:~\$' in cyan and green. A grey cursor block is visible after the prompt.

```
abel@osx:~$ node
> .exit
abel@osx:~$ node
>
(To exit, press ^C again or type .exit)
>
abel@osx:~$ █
```

# Run File

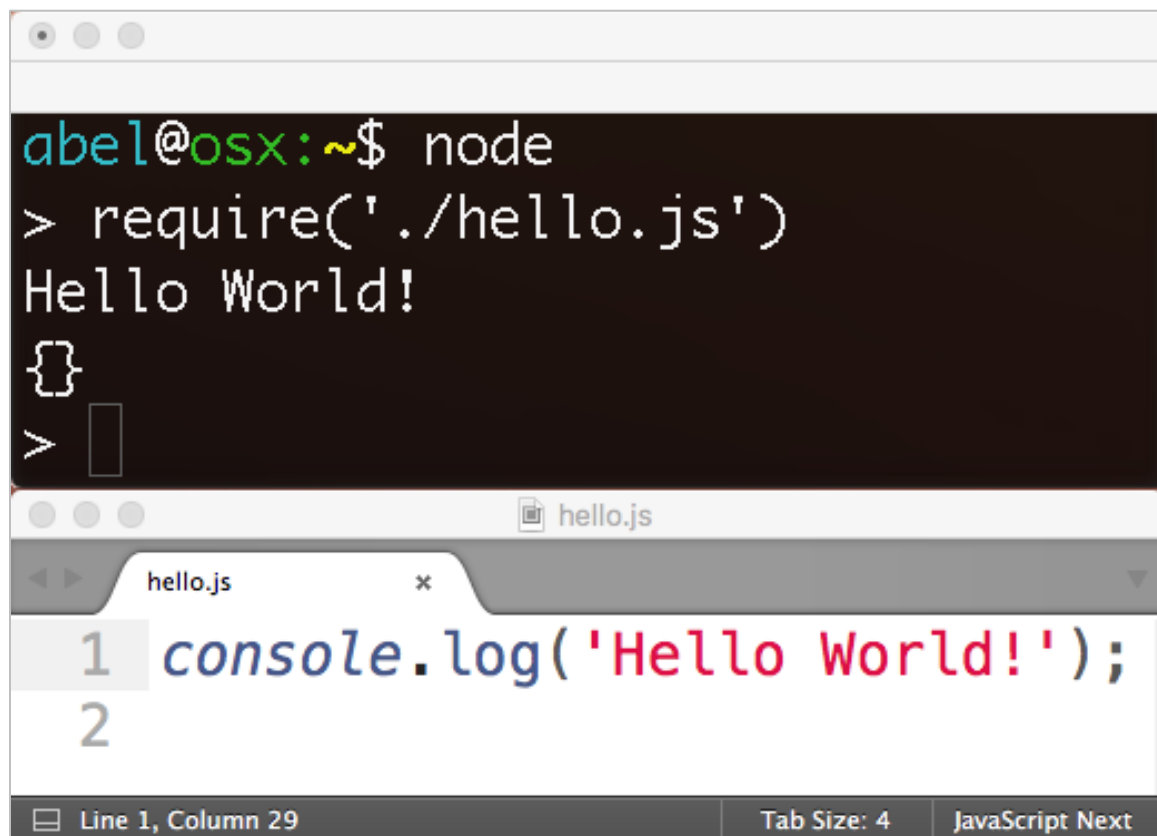


The image shows two overlapping windows. The top window is a 'Node.js command prompt' with a black background and white text. It shows the command `node helloWorld.js` being executed, resulting in the output `Hello world!`. The bottom window is a 'Sublime Text' editor with a white background and a menu bar (File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help). It has a single tab for `helloWorld.js` and displays the following JavaScript code:

```
1 function sayHello(){  
2     console.log('Hello World!');  
3 }  
4 sayHello();
```

The status bar at the bottom indicates 'Line 4, Column 12', 'Tab Size: 4', and 'JavaScript'.

# Load File



The image shows a terminal window and a code editor window. The terminal window displays the command `node` being run, followed by the command `require('./hello.js')`, which outputs `Hello World!`. The code editor window shows the contents of `hello.js`, which is `console.log('Hello World!');`.

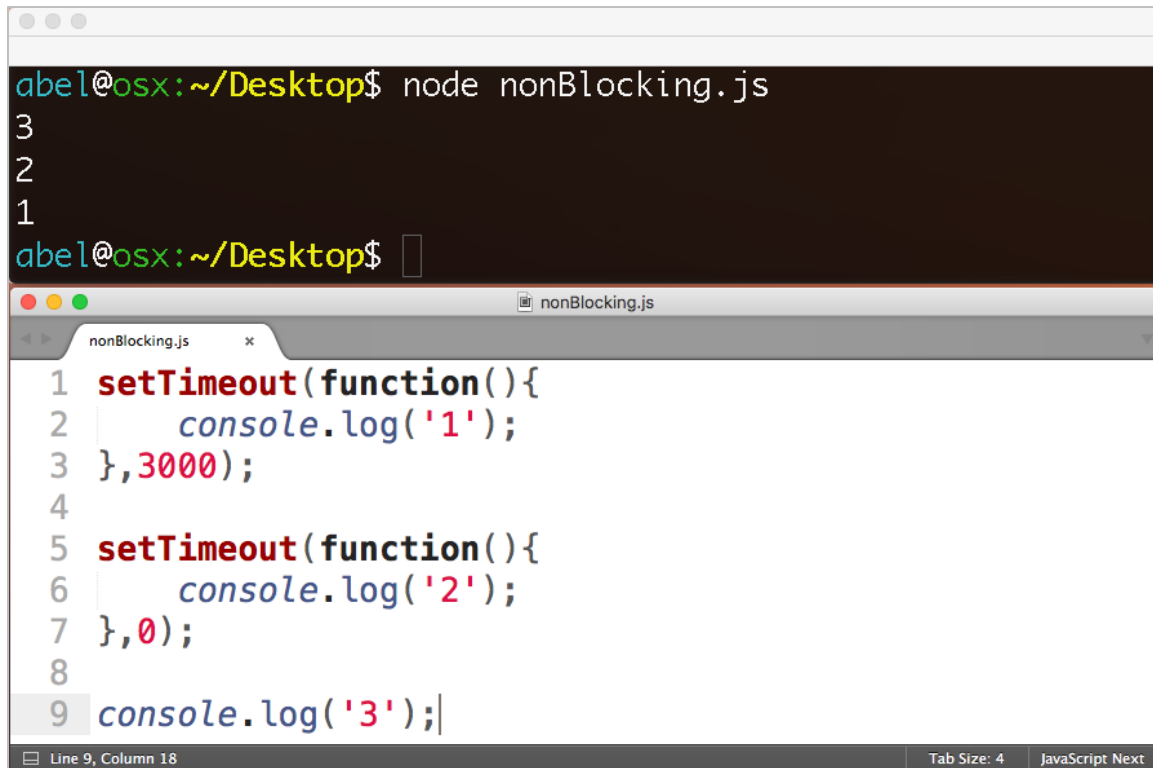
```
abel@osx:~$ node
> require('./hello.js')
Hello World!
{}
>
```

```
hello.js
1 console.log('Hello World!');
2
```

Line 1, Column 29      Tab Size: 4      JavaScript Next



# Non-Blocking

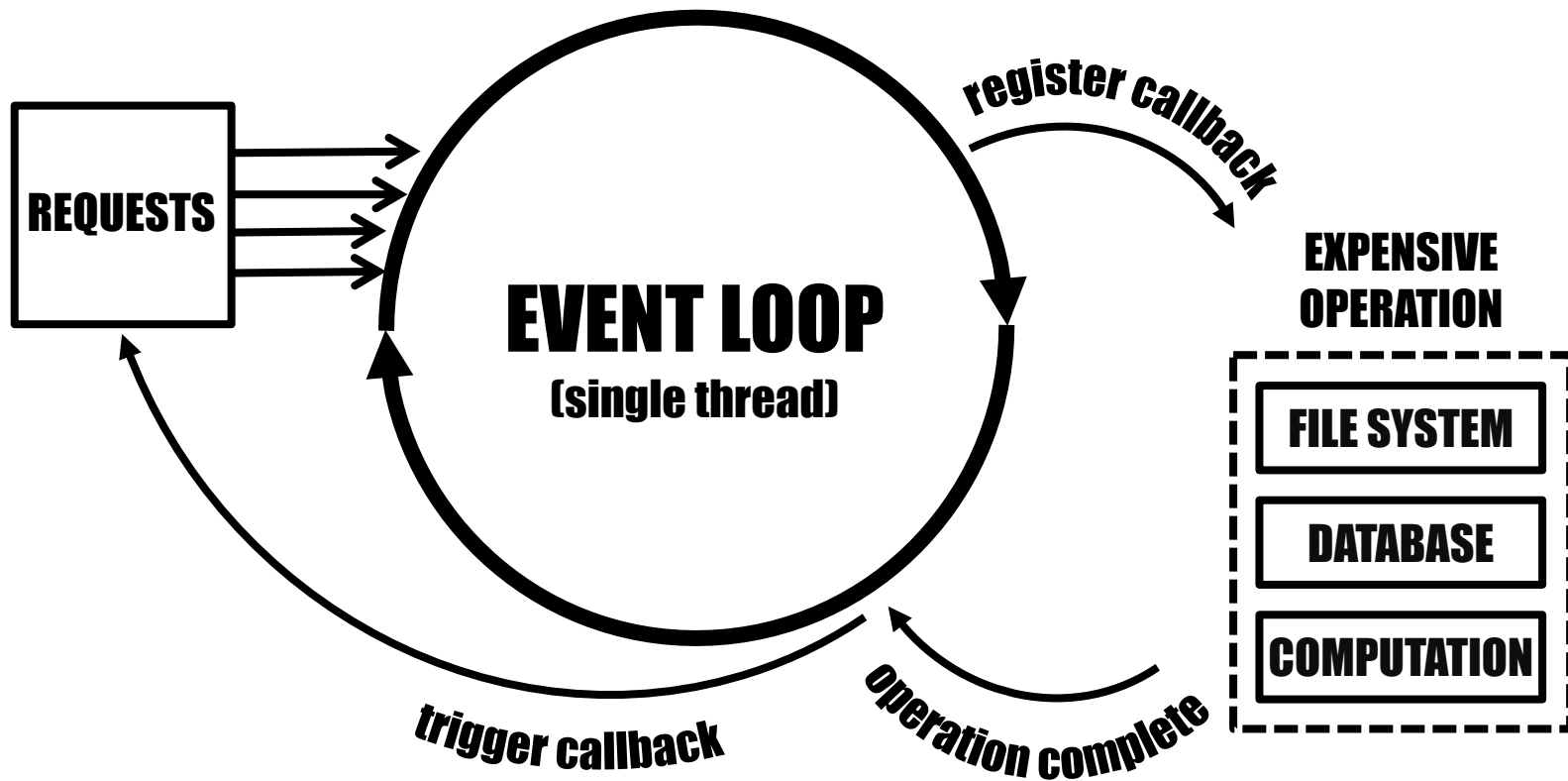


The image shows a terminal window and a code editor. The terminal window, titled 'nonBlocking.js', displays the command `node nonBlocking.js` and the output `3`, `2`, and `1` on separate lines. The code editor, also titled 'nonBlocking.js', shows the following JavaScript code:

```
1 setTimeout(function() {  
2   console.log('1');  
3 }, 3000);  
4  
5 setTimeout(function() {  
6   console.log('2');  
7 }, 0);  
8  
9 console.log('3');
```

The status bar at the bottom indicates 'Line 9, Column 18', 'Tab Size: 4', and 'JavaScript Next'.





# Architectural Shift

**1.old)** web server with application logic

**1.new)** app that can connect and collaborate

**2.old)** stateful

**2.new)** stateless

# Architectural Shift

**1.old)** blocking

**1.new)** non-blocking

**2.old)** process per request

**2.new)** single process



# MODULES

# What is npm?



Package manager. Installs, publishes and manages node programs

What is npm? Largest Ecosystem

**475K Modules**

**2.7 Billion/Week**

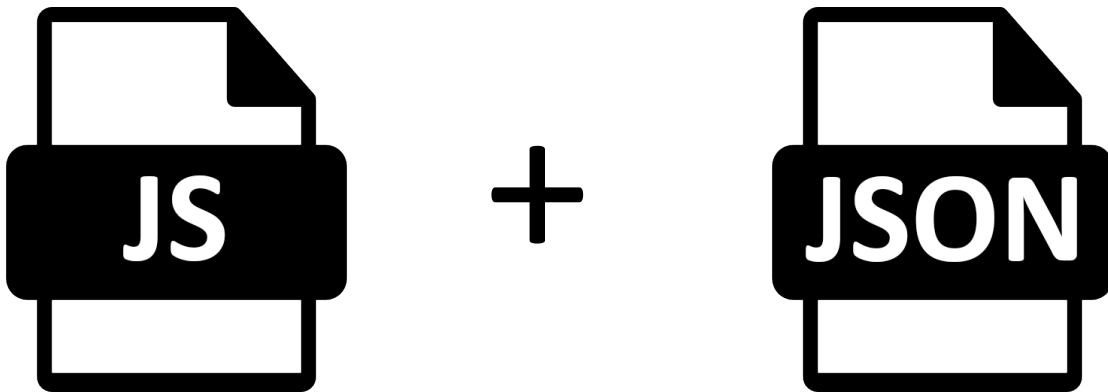


# Advantages

- Small pieces, loosely joined
- Leverage external packages
- Leverage internal packages
- Facilitate collaboration
- Packages are discoverable in npm

# What is a Module?

A Module is some JavaScript  
paired with a package.json file



```
abel@osx:~/Desktop$ node consumeHello.js  
Hello World  
abel@osx:~/Desktop$
```

```
1 var hello = require('./hello.js');  
2  
3 hello();
```



Use module with  
“require”

```
1 module.exports = function(){  
2   console.log('Hello World');  
3 };
```



Make module with  
“module.exports”



Creating node/npm apps

```
abel@osx:~/mywork$ npm init
Press ^C at any time to quit.
name: (mywork)
version: (1.0.0)
description: sample npm app
entry point: (index.js)
test command:
git repository:
keywords:
license: (ISC) MIT
About to write to /Users/abel/mywork/package.json:
```

```
{
  "name": "mywork",
  "version": "1.0.0",
  "description": "sample npm app",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" &&
  },
  "author": "abelsan <abel@mit.edu>",
  "license": "MIT"
}
```

Is this ok? (yes) yes

```
abel@osx:~/mywork$
```

“npm init”  
generates the  
configuration file  
package.json

```
abel@osx:~/mywork$ more package.json
{
  "name": "mywork",
  "version": "1.0.0",
  "description": "sample npm app",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "abelsan <abel@mit.edu>",
  "license": "MIT"
}
abel@osx:~/mywork$
```



adding npm packages  
to your application

```
abel@osx:~/mywork$ npm install request --save
```

Local installation  
of request  
package.  
Dependency  
added to  
package.json

```
abel@osx:~/mywork$ more package.json
{
  "name": "mywork",
  "version": "1.0.0",
  "description": "sample npm app",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "abelsan <abel@mit.edu>",
  "license": "MIT",
  "dependencies": {
    "request": "^2.80.0"
  }
}
```

# Active Learning

- ...