



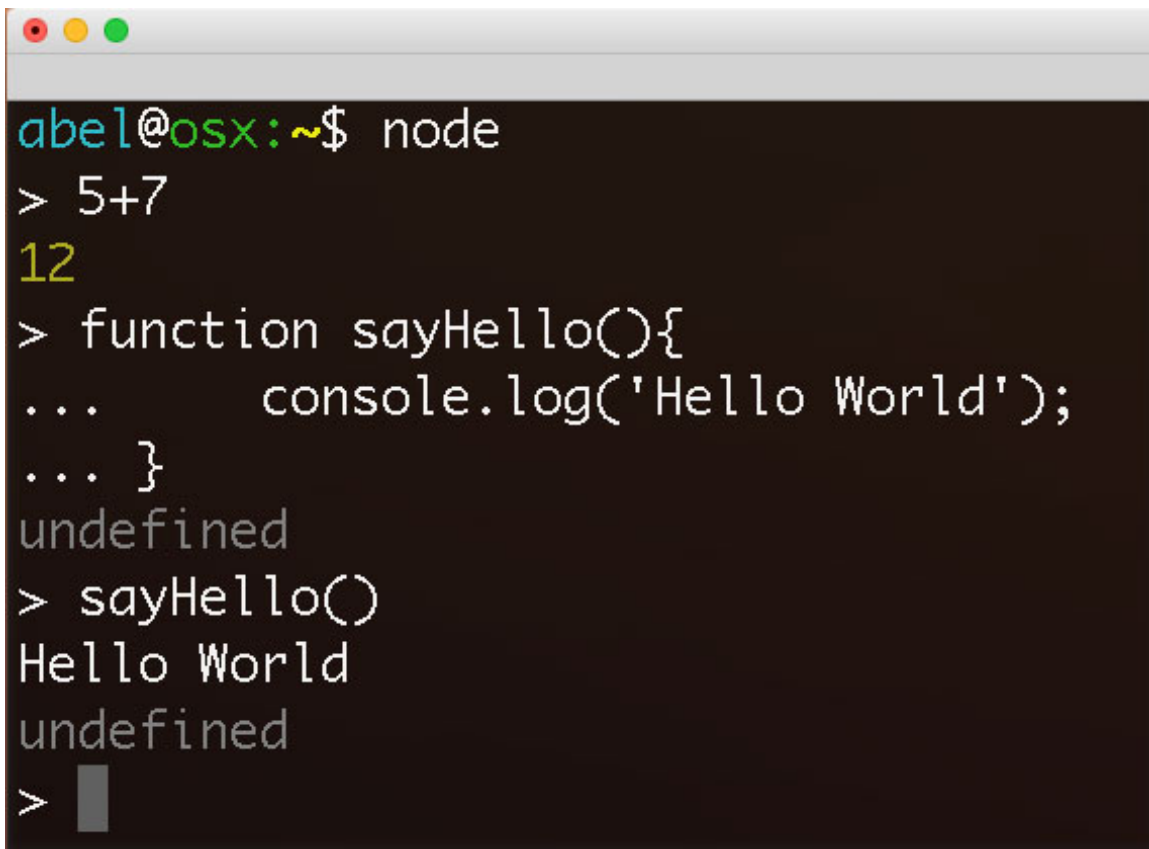


*Node's goal is to provide an easy way
to build scalable network programs*

Node.JS

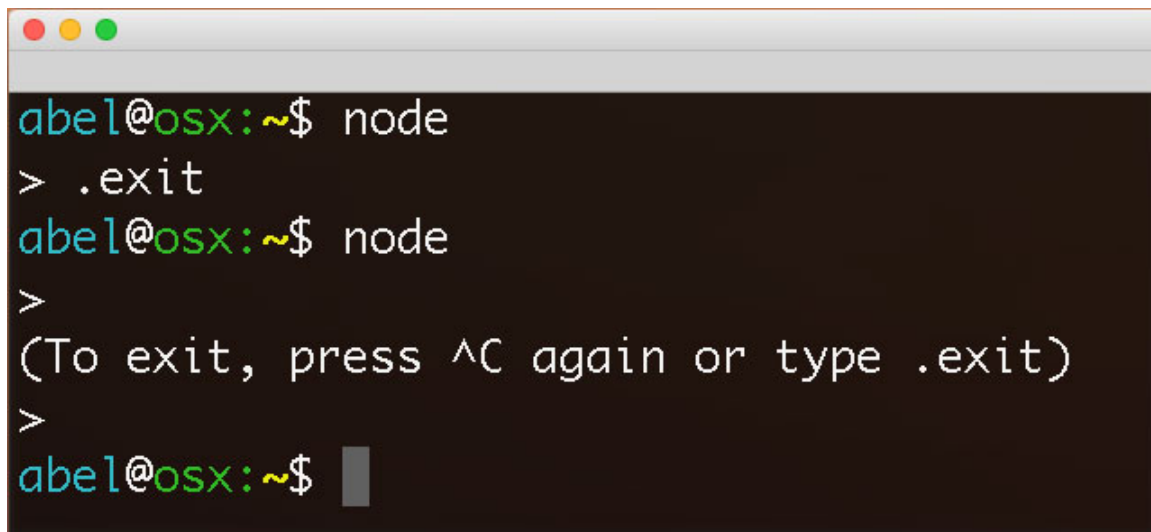
Node.js is an open source, cross-platform, JavaScript runtime built on Chrome's V8 JavaScript engine, for developing server and client side applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.

Console Use

A screenshot of a macOS-style terminal window with a dark background. The window has three colored window control buttons (red, yellow, green) in the top-left corner. The terminal shows a Node.js REPL session. The prompt is 'abel@osx:~\$'. The user enters 'node', which starts the REPL. The prompt changes to '>'. The user enters '5+7', and the output '12' is displayed. The user enters a function definition: 'function sayHello(){ console.log('Hello World'); }'. The output 'undefined' is shown. The user enters 'sayHello()', and the output 'Hello World' is shown. The prompt returns to '>' with a cursor. The text in the terminal is as follows:

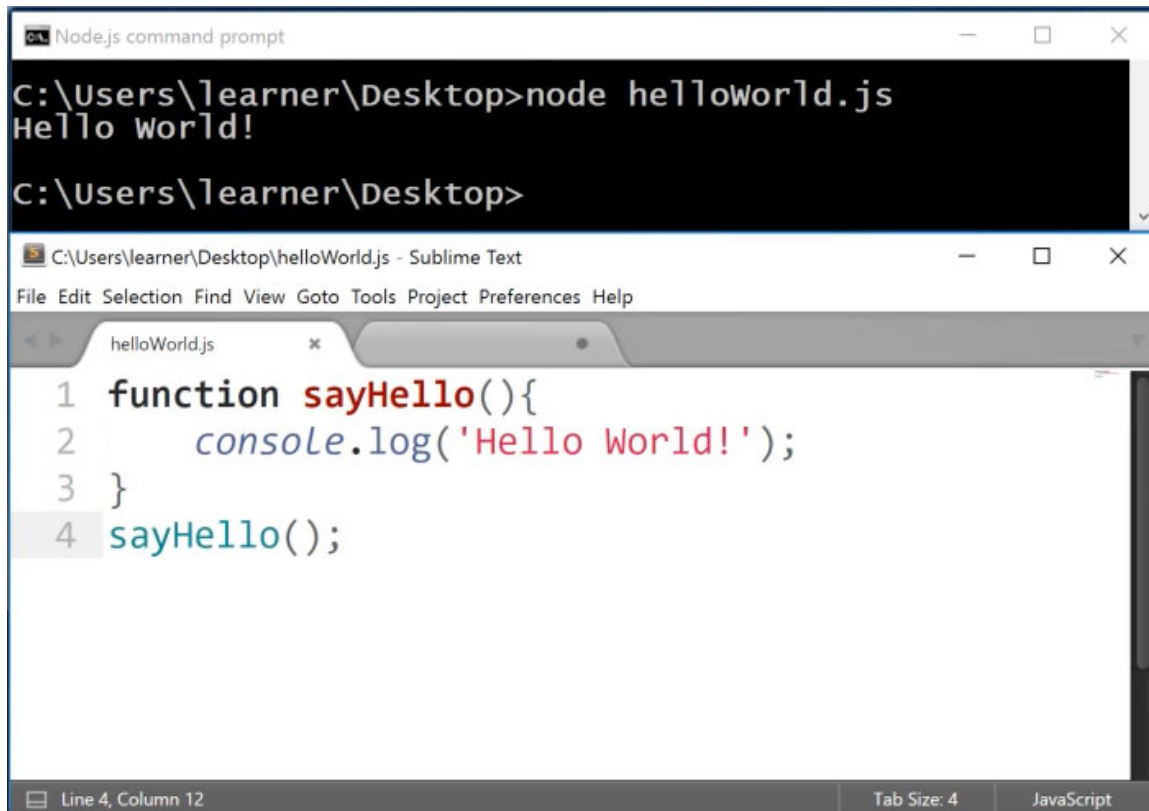
```
abel@osx:~$ node
> 5+7
12
> function sayHello(){
...   console.log('Hello World');
... }
undefined
> sayHello()
Hello World
undefined
>
```

Exit Console

A screenshot of a macOS terminal window with a dark background. The window has a title bar with three colored window control buttons (red, yellow, green) on the left. The terminal text shows a Node.js REPL session. The prompt 'abel@osx:~\$' is 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 returns to '>' in white. The user enters a blank line, and the prompt returns to 'abel@osx:~\$' in cyan and green, followed by a grey cursor bar.

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

Run File

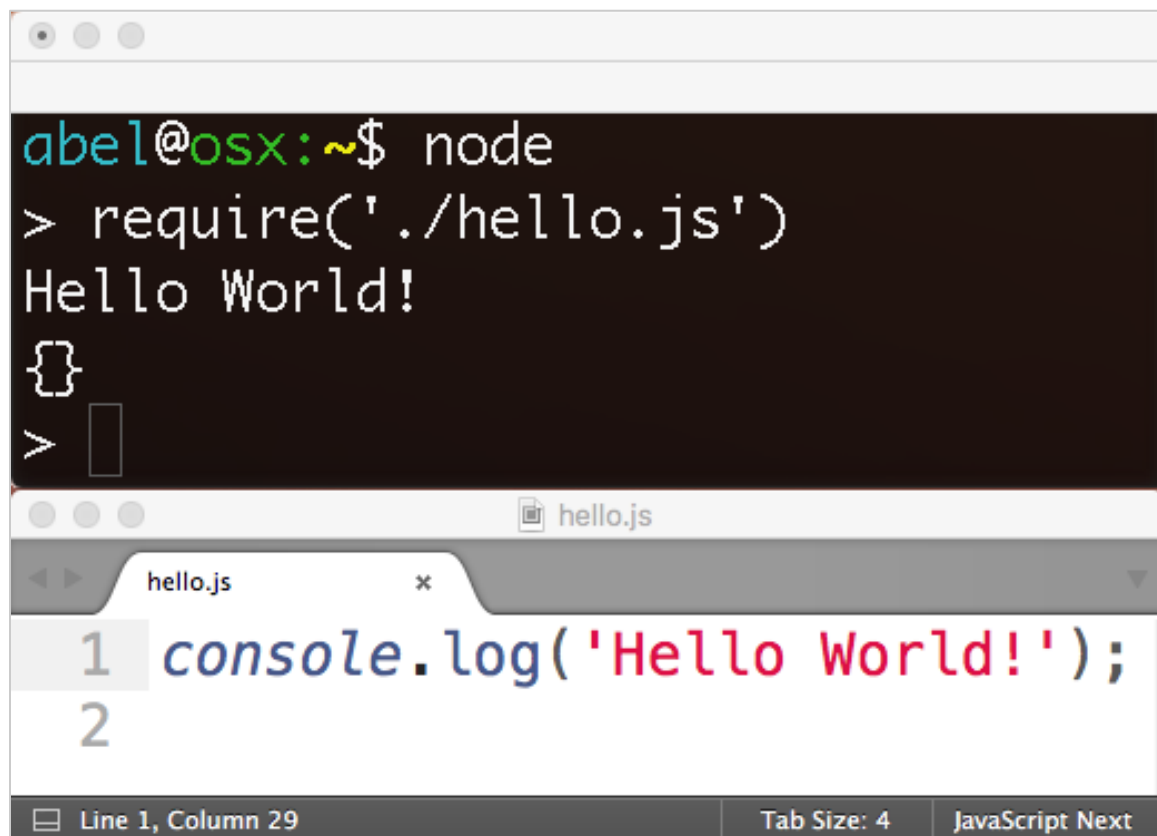


The image shows two overlapping windows. The top window is a 'Node.js command prompt' with a black background and white text. It shows the command `node helloWorld.js` being executed, resulting in the output `Hello world!`. The bottom window is a 'Sublime Text' editor with a white background and a menu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

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

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

Load File



The image shows two overlapping windows. The top window is a terminal with a dark background. It shows a user named 'abel' on an 'osx' machine in the home directory (~) running the 'node' command. The prompt changes to '>' and the user enters 'require('./hello.js')'. The output is 'Hello World!'. The prompt returns to '>' and the user enters a closing curly brace '}'.

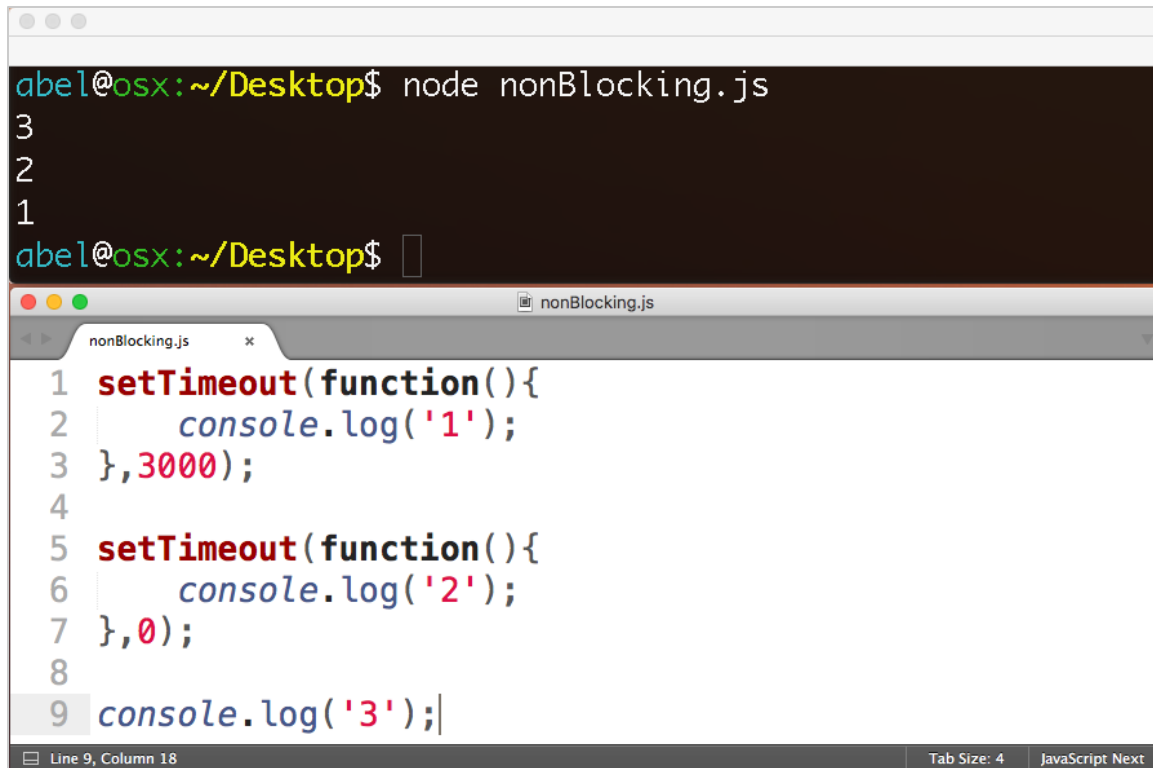
The bottom window is a code editor titled 'hello.js'. It has a tab labeled 'hello.js' with a close button 'x'. The code in the editor is on a single line: `console.log('Hello World!');`. The line number '1' is visible on the left. The status bar at the bottom shows 'Line 1, Column 29', 'Tab Size: 4', and 'JavaScript Next'.

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

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

Line 1, Column 29 Tab Size: 4 JavaScript Next

Non-Blocking

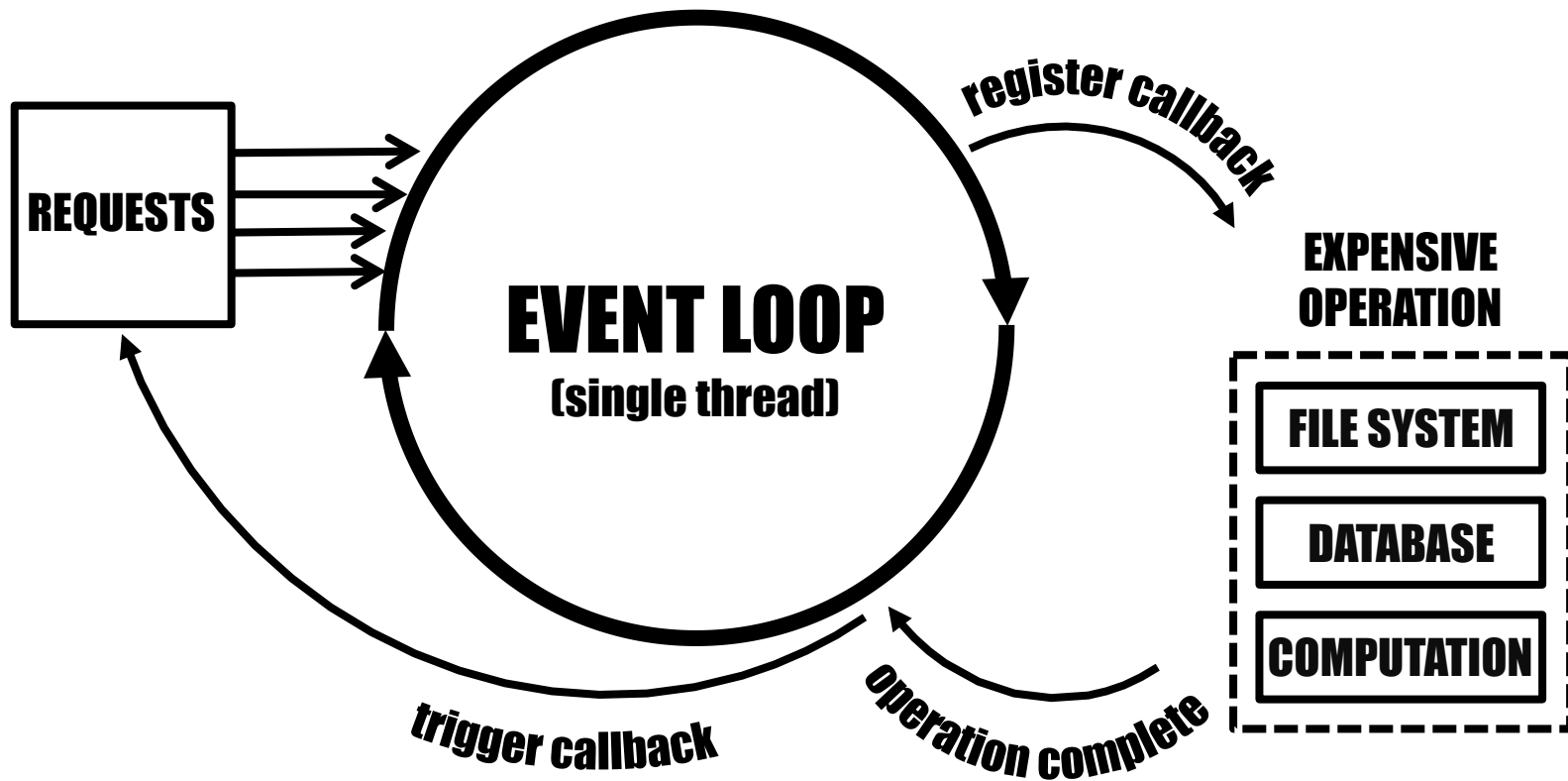


The image shows a terminal window and a code editor. The terminal window, titled 'nonBlocking.js', displays the command `node nonBlocking.js` and its output: `3`, `2`, and `1`. The code editor, also titled 'nonBlocking.js', shows the following JavaScript code:

```
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 some 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

```
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"

What is npm?



Package manager. Installs, publishes and manages node programs

What is npm?

400,000+ Modules

What is npm?

261 Million

downloads in the past day

What is npm?

1.9 Billion

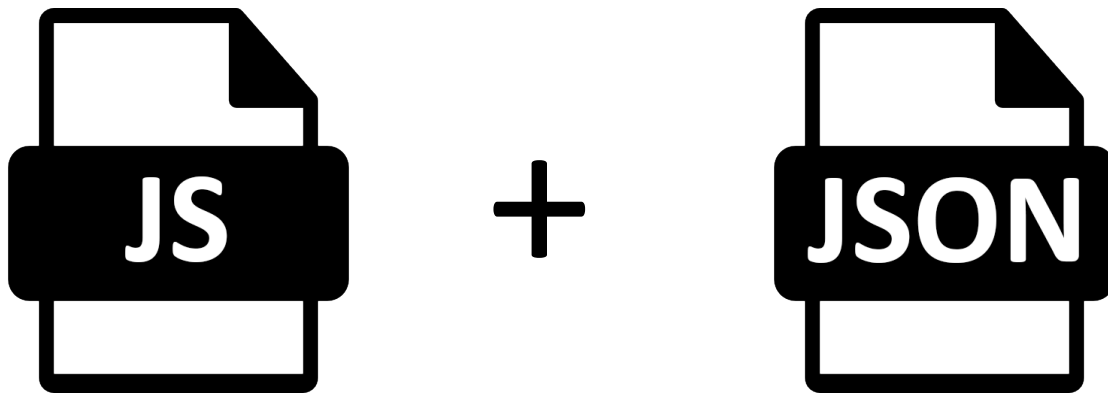
downloads in the past week

What is npm?

7.8 Billion
downloads per month

What is a Module?

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



Advantages

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



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
```

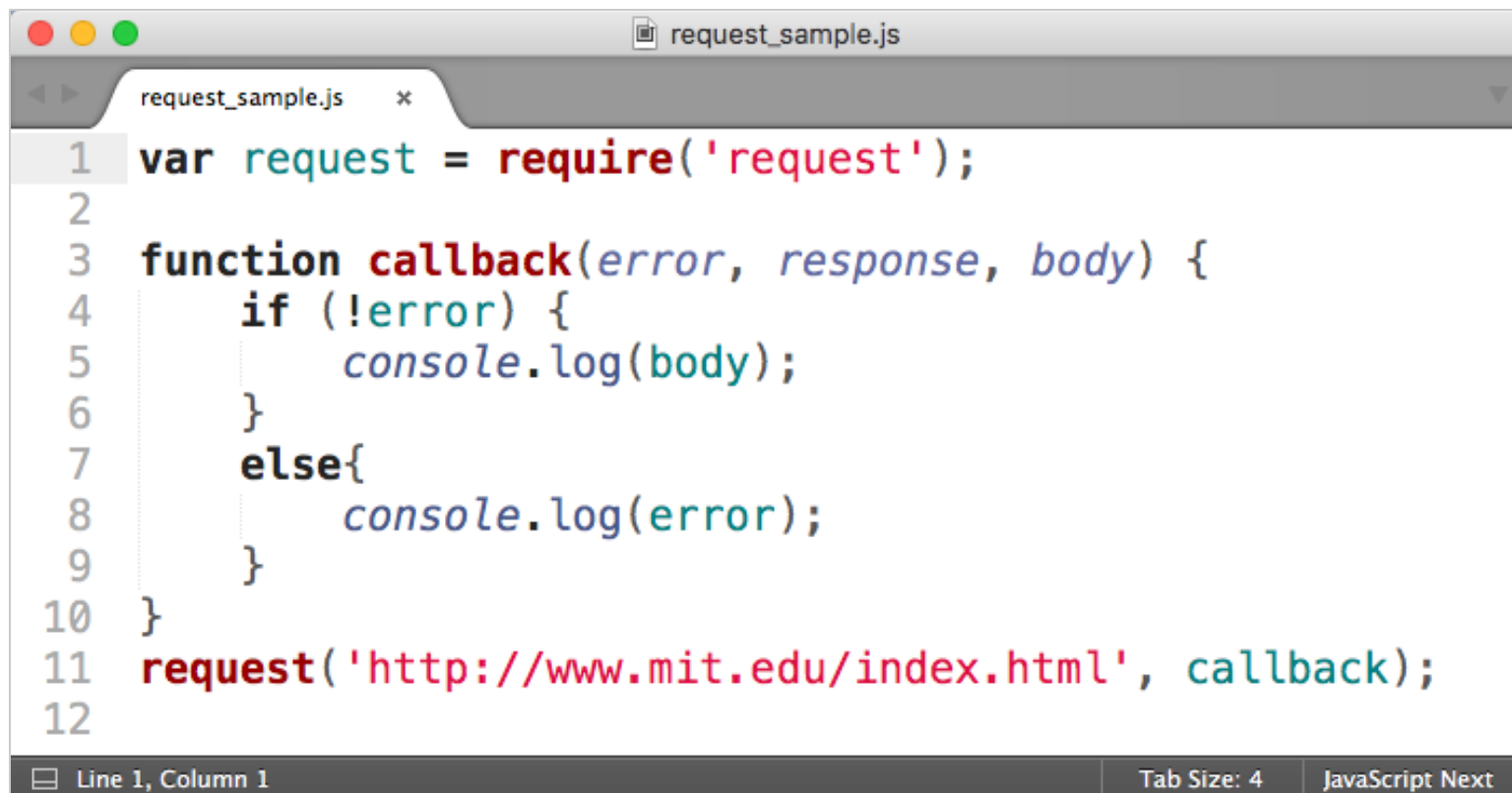
Local installation of
request package.
Dependency added
to package.json

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

Active Learning

- ...

Request Basics

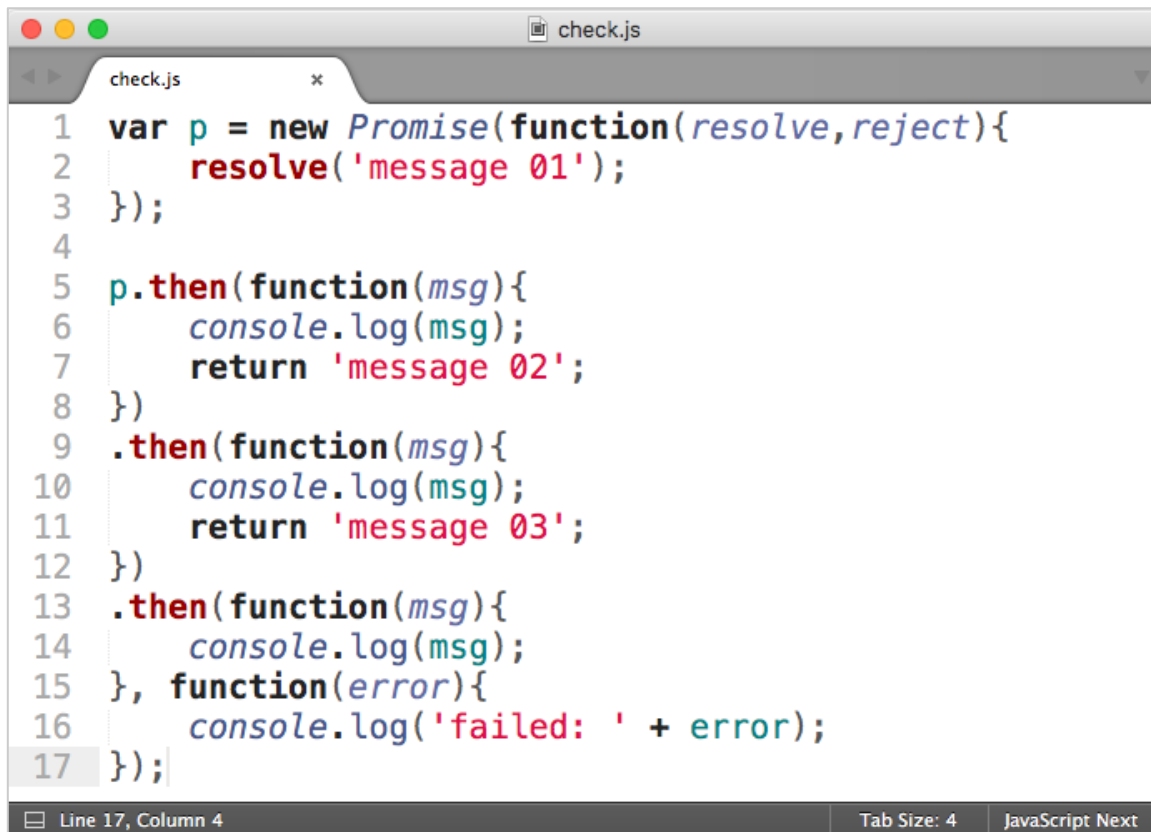


```
1 var request = require('request');
2
3 function callback(error, response, body) {
4     if (!error) {
5         console.log(body);
6     }
7     else{
8         console.log(error);
9     }
10 }
11 request('http://www.mit.edu/index.html', callback);
12
```

Line 1, Column 1 Tab Size: 4 JavaScript Next

Promises Basics

- Basics ...



```
1 var p = new Promise(function(resolve,reject){
2     resolve('message 01');
3 });
4
5 p.then(function(msg){
6     console.log(msg);
7     return 'message 02';
8 })
9 .then(function(msg){
10    console.log(msg);
11    return 'message 03';
12 })
13 .then(function(msg){
14    console.log(msg);
15 }, function(error){
16    console.log('failed: ' + error);
17 });
```

Line 17, Column 4 Tab Size: 4 JavaScript Next