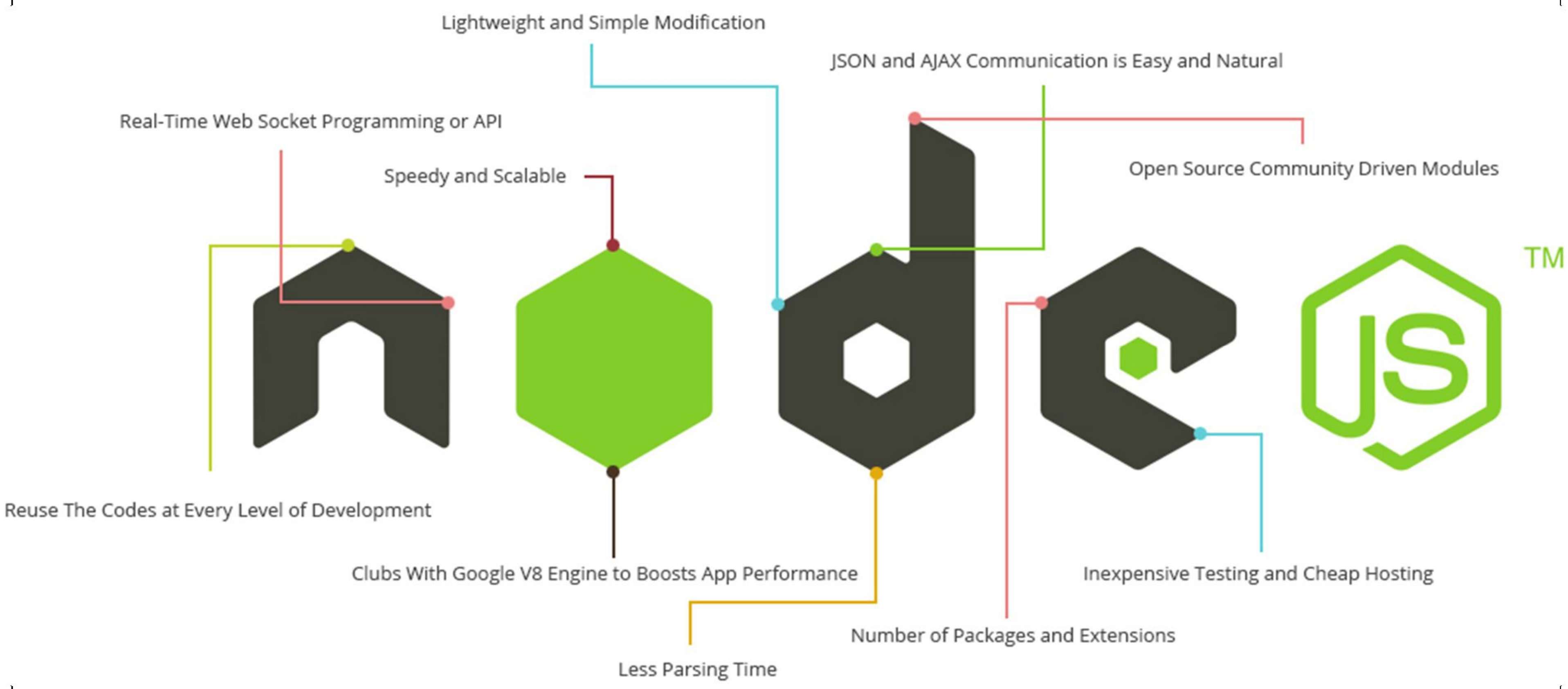
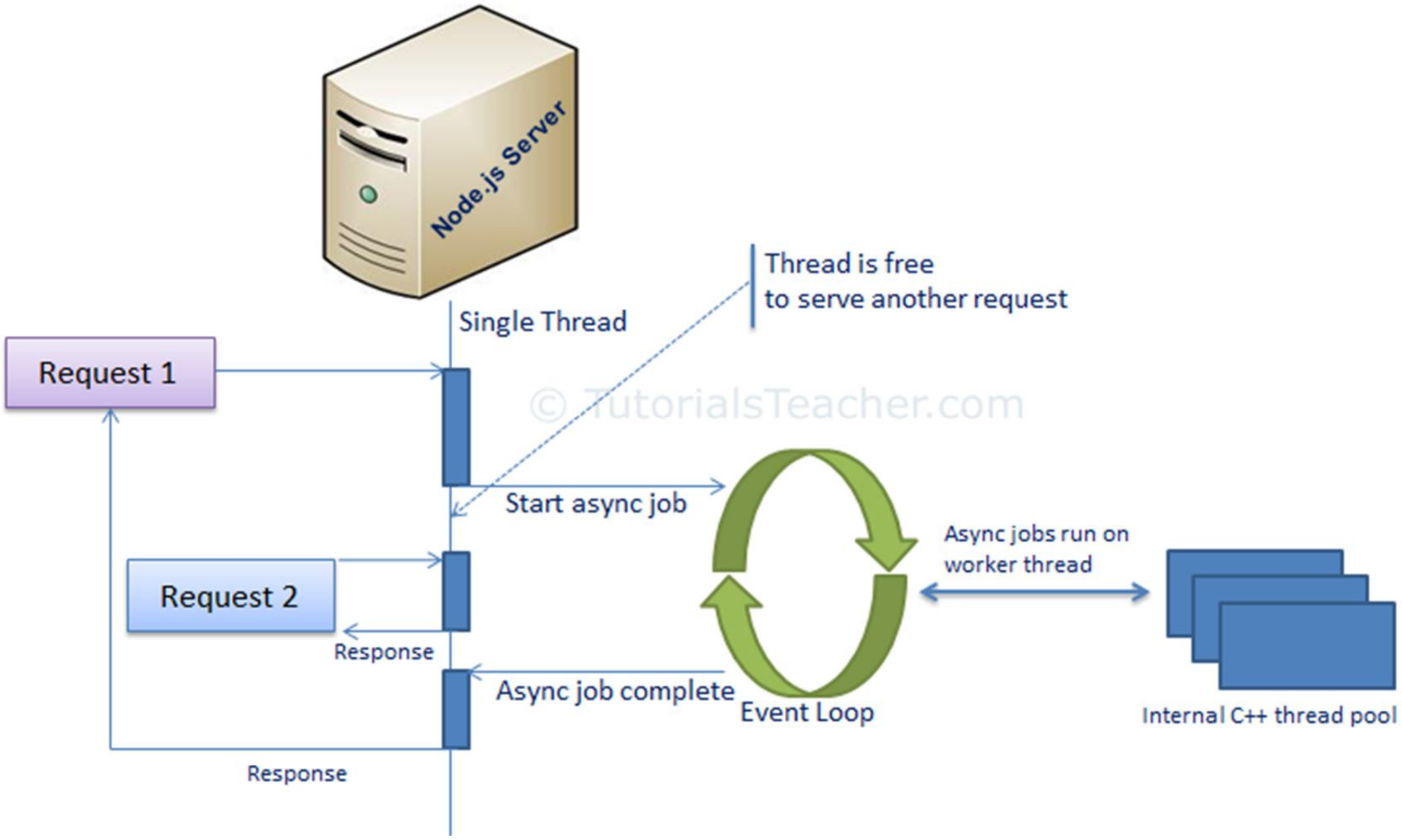
Node JS


Lightweight and Simple Modification
JSON and AJAX Communication is Easy and Natural
Real-Time Web Socket Programming or API
Speedy and Scalable
Open Source Community Driven Modules
TM
Reuse The Codes at Every Level of Development
Clubs With Google V8 Engine to Boosts App Performance
Inexpensive Testing and Cheap Hosting
Number of Packages and Extensions
Less Parsing Time


Node.js Server
Thread is free
to serve another request
Single Thread
Request 1
© TutorialsTeacher.com
Start async job
Async jobs run on
worker thread
Request 2
Response
Async job complete
Event Loop
Internal C++ thread pool
Response

Introduction

- Node.js is a very powerful JavaScript-based framework/platform built on Google Chrome's JavaScript V8 Engine.
- Node.js allows you to run JavaScript on the server.
- Node.js was developed by Ryan Dahl in 2009 and its latest version is v0.10.36.
- It is used to develop I/O intensive web applications like video streaming sites, single-page applications, networking applications, and other web applications.
- Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.
- Node.js is open source, completely free, and used by thousands of developers around the world.

Node.js = Runtime Environment + JavaScript Library

Why Node.js?

- **Node.js uses asynchronous programming!**
- A common task for a web server can be to open a file on the server and return the content to the client.

Here is how PHP or ASP handles a file request:

- Sends the task to the computer's file system.
- Waits while the file system opens and reads the file.
- Returns the content to the client.
- Ready to handle the next request.

Here is how Node.js handles a file request:

- Sends the task to the computer's file system.
- Ready to handle the next request.
- When the file system has opened and read the file, the server returns the content to the client.
- Node.js eliminates the waiting, and simply continues with the next request.
- Node.js runs single-threaded, non-blocking, asynchronously programming, which is very memory efficient.

Who uses Node.js?

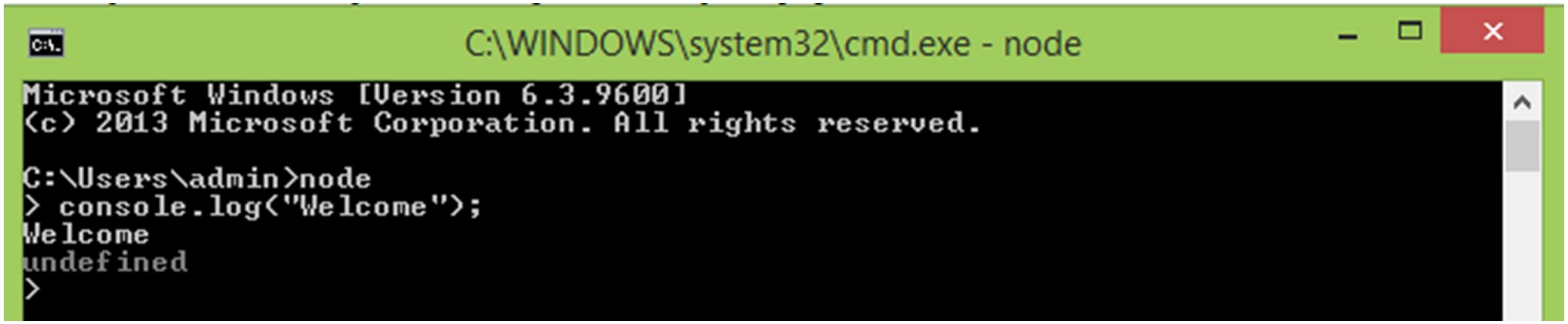
- Following is the link on github wiki containing an exhaustive list of projects, application and companies which are using Node.js. This list includes eBay, General Electric, GoDaddy, Microsoft, PayPal, Uber, Wikipins, Yahoo!, and Yammer to name a few.

What Can Node.js Do?

- Node.js can generate dynamic page content
- Node.js can create, open, read, write, delete, and close files on the server
- Node.js can collect form data
- Node.js can add, delete, modify data in your database

What is a Node.js File?

- Node.js files contain tasks that will be executed on certain events
- A typical event is someone trying to access a port on the server
- Node.js files must be initiated on the server before having any effect
- Node.js files have extension ".js"
- **Download Node.js**
- The official Node.js website has installation instructions for Node.js: https://nodejs.org
- Check the installation by



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\admin>node
> console.log("Welcome");
Welcome
undefined
>
```

Node.js HTTP Module

The Built-in HTTP Module

- Node.js has a built-in module called HTTP, which allows Node.js to transfer data over the Hyper Text Transfer Protocol (HTTP).
- To include the HTTP module, use the require() method:
- var http = require('http');

Node.js as a Web Server

- The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client.
- Use the createServer() method to create an HTTP server.
- The function passed into the http.createServer() method, will be executed when someone tries to access the computer on port 8080.

Add an HTTP Header

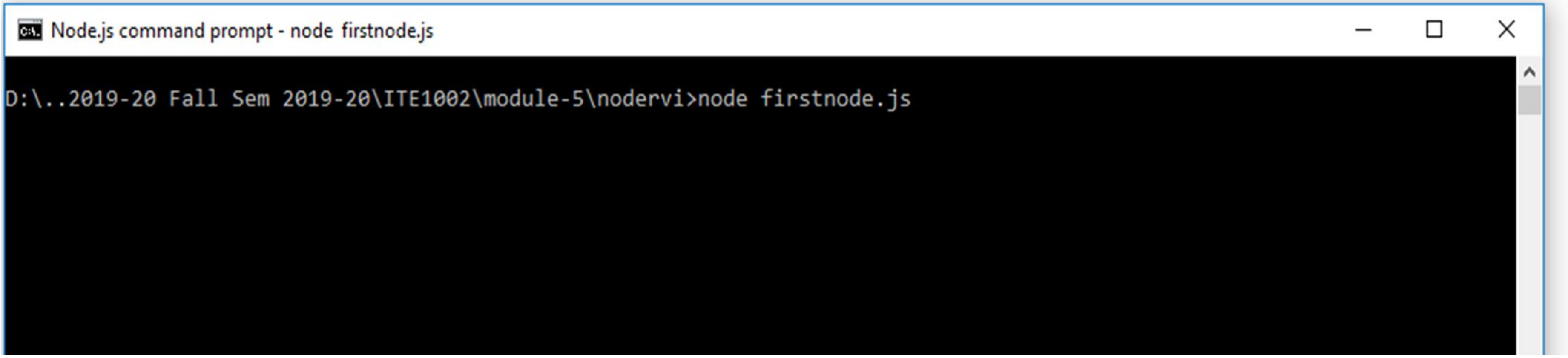
- If the response from the HTTP server is supposed to be displayed as HTML, you should include an HTTP header with the correct content type.
- The first argument of the res.writeHead() method is the status code, 200 means that all is OK, the second argument is an object containing the response headers.

Read the Query String

- The function passed into the http.createServer() has a req argument that represents the request from the client, as an object (http.IncomingMessage object).
- This object has a property called "url" which holds the part of the url that comes after the domain name:

C:\Users\admin\firstnode.js

```
var http = require('http');
http.createServer(function (req, res)
{
   res.writeHead(200,{'Content-Type':'text/html'});
   res.end('Hello vijayan Your Node JS Server is ready!');
}).listen(8080);
```



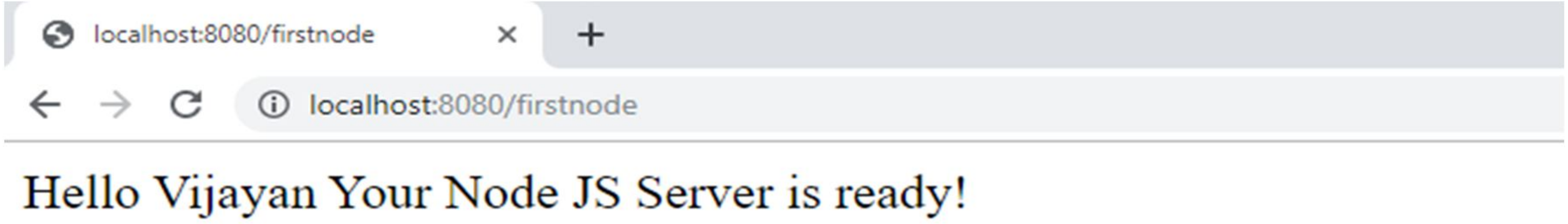
```
D:\..2019-20 Fall Sem 2019-20\ITE1002\module-5\nodervi>node firstnode.js
```

firstnode.js - explanation

- The basic functionality of the "require" function is that it reads a JavaScript file, executes the file, and then proceeds to return an object. Using this object, one can then use the various functionalities available in the module called by the require function. So in our case, since we want to use the functionality of http and we are using the require(http) command.
- In this 2nd line of code, we are creating a server application which is based on a simple function. This function is called, whenever a request is made to our server application.
- When a request is received, we are asking our function to return a 'Hello vijayan Your Node JS Server is ready!' response to the client. The writeHead function is used to send header data to the client and while the end function will close the connection to the client.
- We are then using the server.listen function to make our server application listen to client requests on port no 8080. You can specify any available port over here.

Executing the code

- Save the file on your computer: C:\Users\admin\firstnode.js
- In the command prompt, navigate to the folder where the file is stored. Enter the command node node.js
- Now, your computer works as a server! If anyone tries to access your computer on port 8080, they will get a "Hello vijayan Your Node JS Server is ready!" message in return!
- Start your internet browser, and type in the address: http://localhost:8080



Hello Vijayan Your Node JS Server is ready!