What is node.js?
Node.js is a server-side JavaScript environment that uses an asynchronous event-driven model. This allows Node.js to get excellent performance based on the architectures of many Internet applications.

node.js characteristics

- Implemented on top of the V8 js engine (Chromium and Google Chrome engine).
- Programs are written in plain old javascript, we use the same language for client and server
- Files I/O and db queries are non-blocking
- Great performance even handling a big number of users
- It gets better every time google improves V8
- It has his own package manager called npm (node package manager)

hands on code: http server

```
var http = require("http") ;
var server = http.createServer(function(req, res){
    res.writeHead(200, {"content-type":"text/plain"});
    res.end("Hello World!\n");
}).listen(8000);
```

hands on code: not only http

```
var net = require("net") ;
net.createServer(function(socket){
    socket.on("data", function(data){
      socket.write(data);
    });
}).listen(8000);
```

hands on code: telnet chat

```
var net = require("net");
var clients = [];
net.createServer(function(socket){
  clients.push(socket);
    socket.on("data", function(data){
    for(var i = 0; i < clients.length; i++){
      if(clients[i] == socket) continue;
      clients[i].write(data);
    }
```

```
  });

  socket.on("end", function(){
    var index = clients.indexOf(socket);
    clients.splice(index, 1);
  });
}).listen(8000);
```

How to check it?

telnet localhost 8000

Working on top of node

There are killer frameworks that will make your life easier using node.

- Socket IO
- Express web framework

Socket I/O

Socket.IO is a Node.JS project that makes WebSockets and realtime possible in all browsers. It also enhances WebSockets by providing built-in multiplexing, horizontal scalability, automatic JSON encoding/decoding, and more.

What is NPM?

Node Package Manager (NPM) is a package manager for node

- Command line interface
- Public registry https://www.npmjs.com/
- Allows us to install packages from repo, and publish our own

Understanding package.json

It is a valid JSON object

- name and version fields are required, the combination makes a unique identifier for the package
- Some used fields in package.json
    - description
    - keywords
    - homepage
    - bugs
    - license
    - author & contributors
    - main
    - bin
    - dependencies
    - scripts - {start, preinstall}

Interactive description can be found here: http://browsenpm.org/package.json

Common npm commands

- `npm init` initialize a package.json file
- `npm install <package-name / tarball-file / tarball-url> [-g] [--save] [--save-dev]` install a package, if `-g` option is given package will be installed as a global (irrespective of the directory you installed it from), `--save` and `--save-dev` will add package to your dependencies
- `npm install` install packages listed in package.json (execute this before running the project)
- `npm ls [-g]` listed local packages (without `-g`) or global packages (with `-g`)
- `npm update <package name>` update a package

Express

- Web framework working on top of node.js (http://expressjs.com/)
- High Performance
- Template engines support (jade, ejs)
- CSS engines support (sass, less, stylus)
- Partials support

What Express Does

- Parses arguments and headers
- Routing
- Views
    - Partials
    - Layouts
- Configuration
- Sessions

Hello World app

```
// server.js

var express = require('express')
var app = express()
app.get('/', function (req, res) {
  res.send('Hello World!')
})

var server = app.listen(3000, function () {
  var host = server.address().address
  var port = server.address().port
  console.log('Example app listening at http://%s:%s', host, port)
})
```

Note:
step1: `npm init`
step2: `npm install express --save`

step3: create server.js file
step4: `npm install`
step5: `node server.js`

Basic routing
Routing refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on).
Each route can have one or more handler functions, which is / are executed when the route is matched.
Route definition takes the following structure app.METHOD(PATH, HANDLER), where app is an instance of express, METHOD is an HTTP request method, PATH is a path on the server, and HANDLER is the function executed when the route is matched.

```
// server2.js
      var express = require('express')
      var app = express()

      app.get('/', function (req, res) {
        res.send('Hello World!')
      })

      // accept POST request on the homepage
      app.post('/', function (req, res) {
        res.send('Got a POST request');
      })

      // accept PUT request at /user
      app.put('/user', function (req, res) {
        res.send('Got a PUT request at /user');
      })

      // accept DELETE request at /user
      app.delete('/user', function (req, res) {
        res.send('Got a DELETE request at /user');
      })

      var server = app.listen(3000, function () {

        var host = server.address().address
        var port = server.address().port
```

```
  console.log('Example app listening at http://%s:%s', host,
port)
})
```

How to check it?

curl http://localhost:3000
curl -X POST http://localhost:3000
curl -X PUT http://localhost:3000
curl -X PUT http://localhost:3000/user
curl -X DELETE http://localhost:3000/user

Basic routing (style 2)

```
// server3.js
// Routes
var express = require('express');
var app = express();

require('./routes/site')(app);

var server = app.listen(3000, function () {
  var host = server.address().address;
  var port = server.address().port;
  console.log('Example app listening at http://%s:%s', host, port);
});


// routes/site.js
module.exports = function (app) {
    app.get('/', function (req, res) {
        res.send('Hello World!')
    })

    // accept POST request on the homepage
    app.post('/', function (req, res) {
        res.send('Got a POST request');
    })

    // accept PUT request at /user
    app.put('/user', function (req, res) {
        res.send('Got a PUT request at /user');
```

```
    })

    // accept DELETE request at /user
    app.delete('/user', function (req, res) {
        res.send('Got a DELETE request at /user');
    })
};
```

Basic routing (style 3)

```
// server4.js
var express = require('express');
var app = express();

require('./routes/site')(app);

var server = app.listen(3000, function () {
  var host = server.address().address;
  var port = server.address().port;
  console.log('Example app listening at http://%s:%s', host, port);
});


// routes/site.js
controller = require('../controllers/controller');

module.exports = function (app) {
    app.get('/', controller.doGetAction);
    app.post('/', controller.doPostAction);
    app.put('/user', controller.doPutUserAction);
    app.delete('/user', controller.doDeleteUserAction);
};


// controllers/controller.js
module.exports = {

    doGetAction: function (req, res) {
        res.status(200);
        res.send("AAA");
    },
```

```
    doPostAction: function (req, res) {
        res.status(200);
        res.send("BBB");
    },

    doPutUserAction: function (req, res) {
        res.status(200);
        res.send("CCC");
    },

    doDeleteUserAction: function (req, res) {
        res.status(200);
        res.send("DDD");
    },
};
```