NODE JS

JavaScript on the server

Node Course Guide

This guide is designed to supplement the

Introduction to Node js for beginners + game project Course

It includes source code that follows the lessons of the course. One of the best ways to learn is to try the code out for yourself. Thanks for taking the course, if you have any questions or need clarification on the content please let me know in the Q&A section.

Happy Coding …..

What is Node and how does it work

JavaScript on the server.

- run JavaScript on the server
- open source server environment
- can connect to a database
- Create, open, read write, delete files on your server
- Generate page content

Node.js runs single-threaded, asynchronously programming, which is very memory efficient.

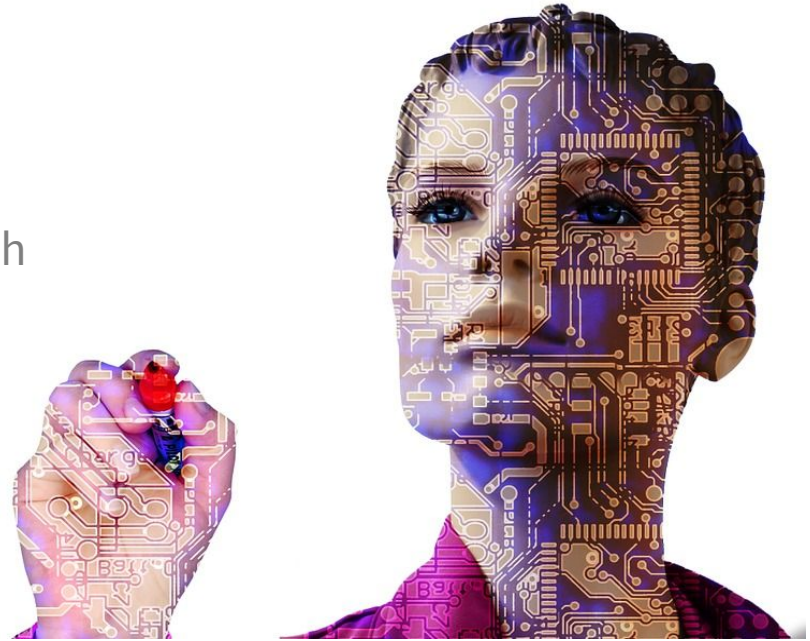
Server-side JavaScript

JavaScript uses browsers to run code on the frontend.

On the backend needs to be interpreted (executed). Node.js uses **Google's V8 VM** which is the same runtime environment the Chrome uses.

Node.js comes with many useful modules - saves from writing from scratch.

Node is runtime environment and a library



Node.js - Install

Node files have a js extension

Download and install node at https://nodejs.org

Terminal - for command line interface

Node package manager - share code and reuse existing code. Faster development.
https://www.npmjs.com/

Install https://www.npmjs.com/get-npm

You will also need an editor - brackets.io

Windows Installer (.msi)	32-bit	64-bit	
Windows Binary (.zip)	32-bit	64-bit	
macOS Installer (.pkg)	64-bit		
macOS Binary (.tar.gz)	64-bit		
Linux Binaries (x64)	64-bit		
Linux Binaries (ARM)	ARMv6	ARMv7	ARMv8
Source Code	node-v10.15.3.tar.gz		

Additional Platforms

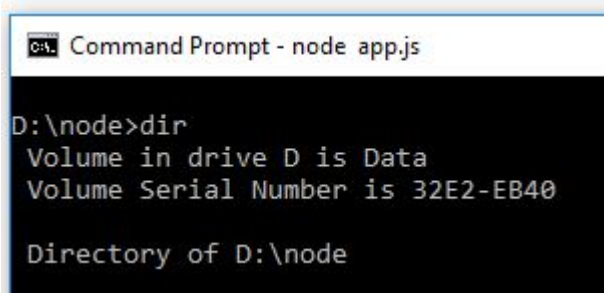
SmartOS Binaries	64-bit
Docker Image	Official Node.js Docker Image
Linux on Power Systems	64-bit
Linux on System z	64-bit
AIX on Power Systems	64-bit

Windows Terminal

Windows - https://cmder.net/ or use the command prompt terminal

Launch the Command Prompt - use the Run window or press the Win + R keys on your keyboard. Then, type cmd and press Enter.

List current directory of files - **dir**
Change directory to D drive - **cd D:**
Change directory down one level - **cd..**
Change directory to folder by name - **cd folder**
Make new folder - **mkdir folderName**
Get help - **help**



Mac Terminal

Open Terminal by pressing Command+Space or select terminal in the applications list.

List current directory of files - **ls**
Change directory to D drive - **cd D:**
Change directory down one level - **cd..**
Change directory to folder by name - **cd folder**
Make new folder - **mkdir folderName**
Get help - **help**

Command Line Launch

One node is installed check to see version installed. Type **node -v**

```
D:\node>node -v
v6.11.3
```

Open your editor and create a js file that contains **console.log('Hello World');** save it as test.js

```
test.js
1  console.log('Hello World');
```

In the terminal type **node test.js** and watch for a result. What gets returned?

```
D:\node>node test.js
Hello World
```

NPM - check if its installed **npm -v** latest version install **npm install npm@latest -g**

```
D:\node>npm -v
6.9.0
```

First Node File

One node is installed check to see version installed. Type **node -v**

```
D:\node>node -v
v6.11.3
```

Open your editor and create a js file that contains **console.log('Hello World');** save it as test.js

```
test.js
1  console.log('Hello World');
```

In the terminal type **node test.js** and watch for a result. What gets returned?

```
D:\node>node test.js
Hello World
```

Node file in the browser

To run the file in a browser you need to serve the web pages. HTTP server

URL where the content will render. Map requests and show pages.

Request handlers on the server side to show page.

Viewing logic to send content to users browser.

NODE can do all this…

First Node HTTP File

Require directive - loads the http module and stores the returned HTTP instance as a variable.

Create Server method. **http.createServer()**

Bind it to port 8080 **.listen(8080)**

Write to the response **writeHead** method. Set status code to 200 success okay and specify content type. **text/plain**

Add body with content **end** method.

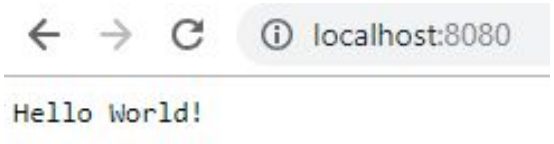
Save the file as **app.js** open the terminal type **node app.js** then open your browser to **http://localhost:8080/**

```
var http = require('http');

http.createServer(function (request, response)
{
  response.writeHead(200, {'Content-Type':
'text/plain'});

  response.end('Hello World!');

}).listen(8080);
```



Try the Code

```
var http = require("http");
http.createServer(function (request, response) {
    response.writeHead(200, {
        "Content-Type": "text/plain"
    });
    response.write("Hello World");
    response.end();
}).listen(8080);
```

IM POSSIBLE

Node How it works

Node uses a single process - where as other languages like PHP would open a new process for every HTTP request. **Callback** is an asynchronous equivalent for a function. (event-driven, asynchronous callbacks allow for moving to the next line.)

```
function message(greeting, callback) {
  console.log(`${greeting}.`);
  callback(); // afterwards gets invoked here
}
function afterwards(){
  console.log('GoodBye!');
}
message('Hello', afterwards);
```

```
D:\node
λ node callback.js
Hello.
GoodBye!
```

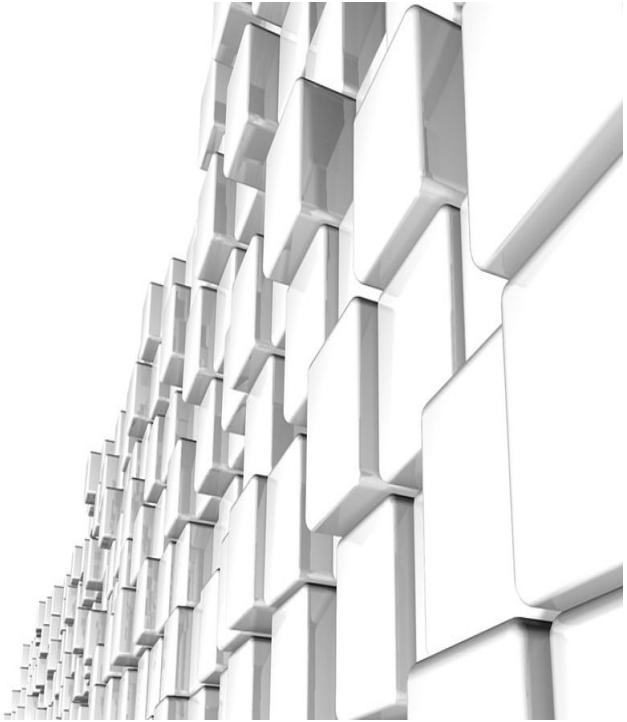
Node modules

Modules in node are like JavaScript libraries. Pre Written code that you can pull into your project and use.

Node.js has a set of built-in modules which you can use without any additional installation.

To include a module, use the require() function with the name of the module.

var http = **require**('http');

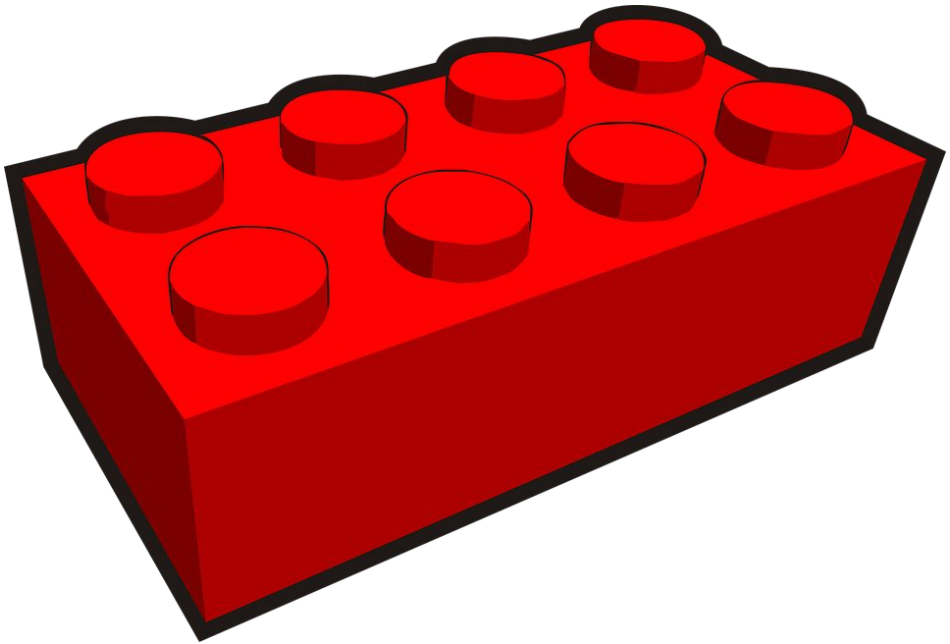


Module

A module is a JavaScript library/file that you can import into other code using Node's require() function

https://www.npmjs.com/

https://www.npmjs.com/search?q=twitter



Create a module

Allows you to easily reuse code.

```
//file called module.js
exports.greeting = "hello world";
```

```
//file called testmod.js
const output = require('./module.js')
console.log(output.greeting);
```

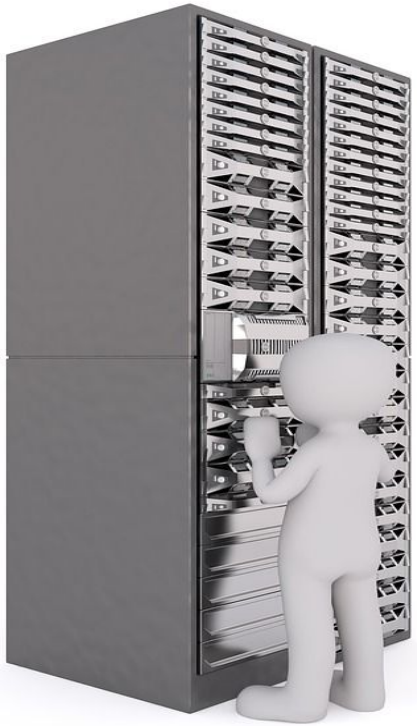
```
D:\node
λ node testmod.js
hello world
```

Node web server

```
const http = require("http");  //loads module called http
function req(request, response) { //callback function
    response.writeHead(200, { //sends HTTP status 200
        "Content-Type": "text/plain"
    });
    response.write("Hello World"); //page content response body
    response.end(); //finish the response.
}

http.createServer(req).listen(8080); //method creates an HTTP server

console.log('Server started');
```



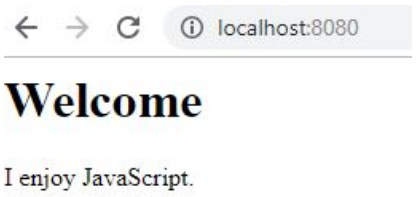
Files on your server

Read index1.html to your web browser.

```
const http = require('http');
const fs = require('fs');

function req(request, response) {
    fs.readFile('index1.html', function (error, data) {
        response.writeHead(200, {
            'Content-Type': 'text/html'
        });
        response.write(data);
        response.end();
    })
}
http.createServer(req).listen(8080);
console.log('Server started');
```

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<h1>Welcome</h1>
<p>I enjoy JavaScript.</p>
</body>
</html>
```

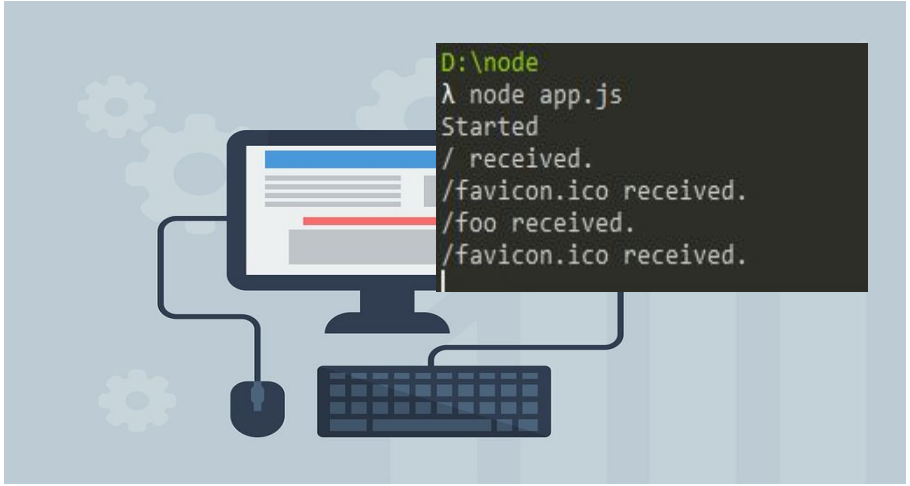


Node URL pathname request

```
const http = require('http');
const url = require('url');
function onRequest(request, response) {
    const base = url.parse(request.url);
    const pathname = base.pathname;
    console.log(pathname + ' received.');
    response.writeHead(200, {
        'Content-Type': 'text/plain'
    });
    response.write('Welcome');
    response.end();
}
http.createServer(onRequest).listen(8080);
console.log('Started');
```

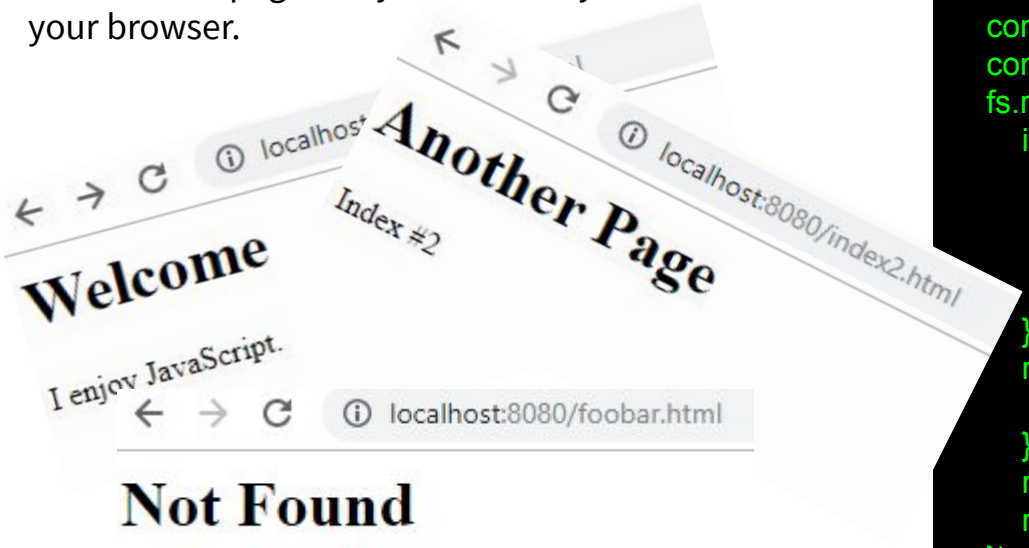
http://localhost:8080/foo

Default is to look for favicon on every page.



Node Routing

Create some pages on your directory. Load them in your browser.




```
const http = require('http');
const url = require('url');
const fs = require('fs');

function onRequest(request, response) {
    const base = url.parse(request.url);
    const fileName = '.' + base.pathname;
    fs.readFile(fileName, function (error, data) {
        if (error) {
            response.writeHead(404, {
                'Content-Type': 'text/html'
            });
            return response.end('<h1>Not Found</h1>');
        }
        response.writeHead(200, {
            'Content-Type': 'text/html'
        });
        response.write(data);
        return response.end();
    })
}
http.createServer(onRequest).listen(8080);
console.log('Started');
```

File Write and Read

Node makes it easy to write and read files from your computer. You can also read directories.
readFile use utf-8 as the character code
readdir - to read the directory file names
writeFile - to write to or create a file.

```
  'package.json',
  'test.js',
  'testmod.js' ]
Hello World
file created
```

```
const fs = require('fs');
fs.readFile('./db.json', 'utf-8', function (err, data) {
    const d = JSON.parse(data); //data from db.json
    console.log(d.message);
})
fs.readdir('./', function (err, data) {
    console.log(data); // all directories from folder
})
const temp = {
    greeting: 'Welcome to Node' //js object
}
fs.writeFile('db1.json', JSON.stringify(temp), function (error) {
//stringify to JSON
    console.log('file created');
})
```

```
{"message":"Hello World"}
```

Loading Packages NPM

Popular package called lodash
https://www.npmjs.com/package/lodash
npm install lodash

- Installs a new folder with all dependant files.
- Warnings if no package.json which lists dependant packages

package.json - Specifics of npm's package.json handling. At the heart of node system all metadata for the project making it easier to package and redistribute.

You can create a package.json file **npm init**
Answer the questions to complete

Short format is **npm init --yes**

```
D:\node
λ npm install lodash
```

```
λ npm install lodash
npm WARN saveError ENOENT: no such file or directory, open 'D:\node\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'D:\node\package.json'
npm WARN node No description
npm WARN node No repository field.
npm WARN node No README data
npm WARN node No license field.

+ lodash@4.17.11
added 1 package from 2 contributors and audited 1 package in 2.128s
found 0 vulnerabilities


D:\node
λ
```

Lodash Quick

https://lodash.com/ - https://lodash.com/docs/4.17.11
First setup Lodash if you haven't already.
npm init - to create package.json file in folder
npm install lodash - to install package
-g to install current package as a global package
(**npm i -g *npm***) shorthand i is for install then last parameter is the package name.
npm install - will install all modules listed as dependencies in package.json .

```
const _ = require('lodash');
let randomNum1 = _.random(10); // 0-10 random number
let randomNum2 = _.random(1, 100); // 0-10 random number
console.log(randomNum1);
console.log(randomNum2);
let myArr = ['apple', 'orange', 'banana', 'blue berry', 'grape',
'cherry'];
let randomItem = _.sample(myArr);
console.log(randomItem);
console.log(myArr);
console.log(_.shuffle(myArr));
console.log(_.shuffle(myArr));
```

```
let counter = 0;
_.times(5, function () { //Loops 5 times
    counter++;
    console.log(counter);
})
let arr2 = _.map(myArr, function (item) {
    console.log(item);
    return item.toUpperCase();
})
console.log(arr2);
console.log(_.map(myArr, _.camelCase));
console.log(_.map(myArr, _.capitalize));
console.log(_.map(myArr, _.upperCase));
console.log(_.map(myArr, _.lowerCase));
//return items that start with
let arr3 = _.map(myArr, function (e) {
    return _.startsWith(e, 'b');
})
console.log(arr3);
let arr4 = _.map(myArr, function (e) {
    return _.endsWith(e, 'e');
})
console.log(arr4);
myArr.forEach(function (e) {
    if (_.endsWith(e, 'e')) {
        console.log(e);
    }
});
```

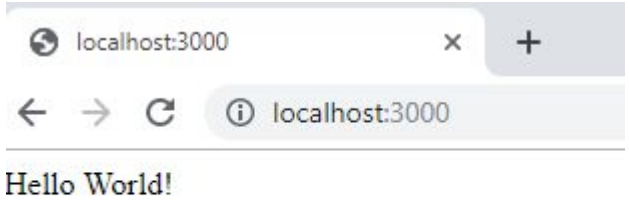
Packages Express

Express is the most popular Node web framework, and is the underlying library for a number of other popular Node web frameworks. Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.
https://expressjs.com/

npm install express --save

http://localhost:3000/

```
npm install express --save
```




```
const express = require('express');
const app = express();
const port = 3000;
app.get('/', function (req, res) {
  res.send('Hello World!');
});
app.get('/about', function(req, res) {
  res.send('About Page');
});
app.listen(port, function () {
  return console.log("Port is ".concat(port, "!"));
});
```

Express static files

__dirname is the directory that will be used.

http://localhost:3000/index1.html

http://localhost:3000

```
const express = require('express');
const app = express();
const port = 3000;
app.use(express.static(__dirname));  // change directory to root

app.listen(port, function () {
  return console.log("Port is "+port + ' '+ __dirname);
});
```

http://localhost:3000/2

http://localhost:3000

```
const express = require('express');
const app = express();
const port = 3000;
app.get('/', function (req, res) {
    res.sendFile(__dirname +'/index1.html');
});
app.get('/2', function (req, res) {
    res.sendFile(__dirname +'/index2.html');
});
app.listen(port, function () {
  return console.log("Port is "+port + ' '+ __dirname);
});
```

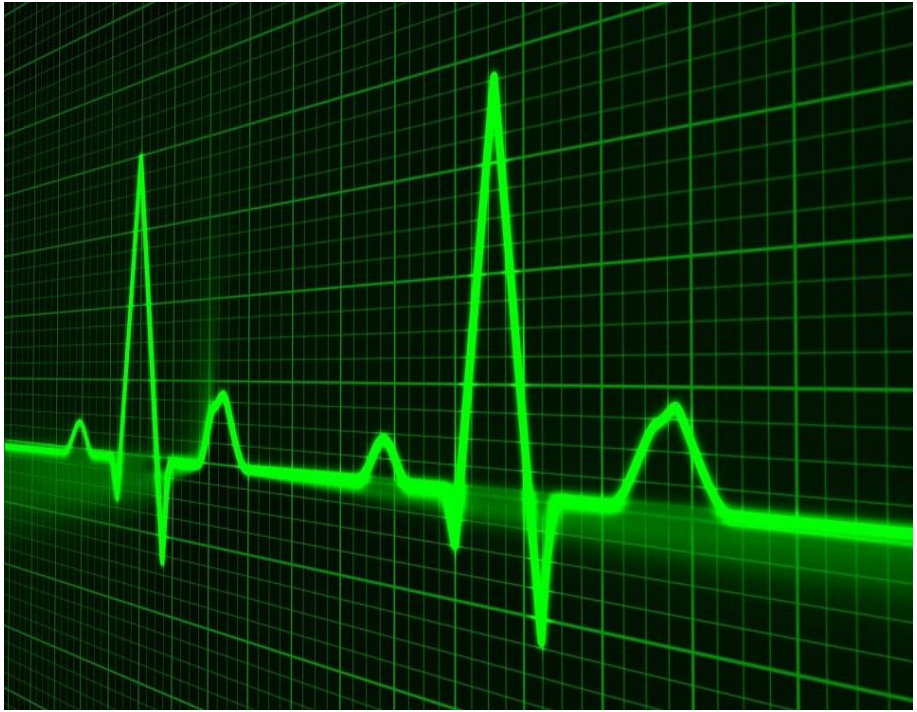
NodeMon

Nodemon is a utility that will monitor for any changes in your source and automatically restart your server. Perfect for development.

npm install -g nodemon

https://nodemon.io/

Run it
 nodemon app.js
** make some changes no more node restart typing ;)



Get data

```
<button>Click</button>
<div class="output"></div>
<script>
    const btn = document.querySelector('button');
    const output = document.querySelector('.output');
    btn.addEventListener('click', listUsers);
    function listUsers() {
        fetch('users').then(function (response) {
            return response.json();
        }).then(function (data) {
            output.innerHTML = "Users<br>";
            data.forEach(function (el) {
                let div = document.createElement('div');
                div.textContent = el.user + ' ' + el.pass;
                output.appendChild(div);
            })
        });
    }
</script>
```

```
const express = require('express');
const app = express();
app.use(express.static(__dirname));

const users = [{
  user: 'Laurence',
  pass: 'secret'
}, {
  user: 'Jane',
  pass: 'password'
}];
app.get('/users', function (req, res) {
  res.send(users);
});
const server = app.listen(3000, function () {
  console.log('server' + server.address().port);
});
```

Body-parser

Parse incoming request bodies in a middleware before your handlers

npm install body-parser

https://www.npmjs.com/package/body-parser

Send data from the front end code form to the backend.

```
<form id="myform" action="login" method="post">
    <input type="text" name="user" value="User Name">
    <input type="text" name="pass" value="Password"> </form>
<div class="btn">Click Me</div>
<script>
    const btn = document.querySelector('.btn');
    btn.addEventListener('click', function () {
        const formData = new FormData(document.forms[0]);
        const searchParams = new URLSearchParams();
        for (const val of formData) {
            console.log(val);
            searchParams.append(val[0], val[1]);
        }
        fetch('/login', {
            method: 'post'
            , body: searchParams
        }).then(function (response) {
            return response.json()
        }).then(function (data) {
            return console.log(data);
        })
    })
</script>
```

Body-parser

Middleware to get data

```
const express = require('express');
const app = express();
const port = 3000;
const bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({
    extended: true
}));
app.use(bodyParser.json()); // for parsing application/json
app.use(express.static(__dirname + '/public'));
app.get('/', function (req, res) {
    res.sendfile("index.html");
});
app.post('/login', function (req, res) {
    let userName = req.body.user;
    let password = req.body.pass;
    console.log("UserName = " + userName + ", password = " +
password);
    res.json({
        status: true
    });
});
app.listen(3000, function () {
    console.log("PORT " + port);
})
```

Socket.io

https://socket.io/

```
const express = require('express');
const app = express();
const port = 3000;
const http = require('http').Server(app);
const socket = require('socket.io')(http);
app.use(express.static(__dirname + '/public'));
app.get('/', function (req, res) {
    res.sendfile("index.html");
});
socket.on('connection', function (s) {
    console.log('user ready');
})
const server = http.listen(port, function () {
    console.log("Ready " + port);
})
```

```
<input name="user" type="text" value="Laurence">
<button class="btn">Roll</button>
<script src="/socket.io/socket.io.js"></script>
<script>
    const socket = io(); //initialization of socket connection
    const btn = document.querySelector('.btn');
    btn.addEventListener('click', function () {

    })
</script>
```

Socket.io send message to users

index.html

https://socket.io/docs/emit-cheatsheet/

http://localhost:3000/players - list players
http://localhost:3000/ - join game

```
<div class="output"></div><input name="user" type="text"
value="Laurence"><button class="btn1">Join</button>
<script src="/socket.io/socket.io.js"></script>
<script>
    const socket = io();
    const output = document.querySelector('.output');
    const btn1 = document.querySelector('.btn1');
    const user = document.querySelector('input[name=user]');
    btn1.addEventListener('click', function () {
        var id = 'player_' + Math.floor(Date.now() * Math.random());
        socket.emit('new player', id, user.value);
        user.style.display = 'none';
        btn1.style.display = 'none';
    })
    socket.on('players', listPlayers);
    function listPlayers(players) {
        output.innerHTML = 'Players<br>';
        players.forEach(function (player) {
            let div = document.createElement('div');
            div.textContent = player.name;
            output.appendChild(div);
        })
    }
</script>
```

app.js

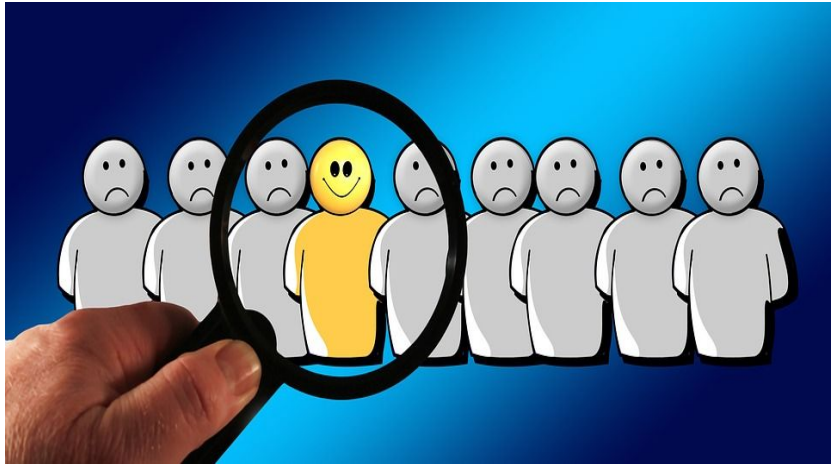
```
const express = require('express');
const app = express();
const port = process.env.PORT || 3000;
const http = require('http').Server(app);
const io = require('socket.io')(http);
let players = [];
const bodyParser = require('body-parser');
app.use(bodyParser.json())
app.use(bodyParser.urlencoded({
    extended: true
}));
app.use(express.static(__dirname + '/public'));
http.listen(port, function () {
    console.log("Ready " + port);
})
app.get('/', function (req, res) {
    res.sendfile("index.html");
});
app.get('/players', function (req, res) {
    res.send(players);
});
```

```
io.on('connection', function (socket) {
    let userId;
    socket.on('disconnect', function (reason) {
        console.log(reason);
        players = players.filter(function (obj) {
            return obj.id !== userId;
        });
        console.log('player left '+userId);
        io.emit('players', players);
    });
    socket.on('new player', function (id, name) {
        userId = id;
        players.push({
            name: name
            , id: id
            , round: 0
            , roll: null
        });
        io.emit('players', players);
    });
    io.emit('players', players);
});
```

Uuid create a User ID

npm install uuid
https://www.npmjs.com/package/uuid

Creates a unique id for every connection.



```
const port = 3000;
const http = require('http')
const express = require('express');
const app = express();
const server = http.createServer(app);
const uuidv1 = require('uuid/v1');
server.listen(port);
console.log('Listening on port ' + port);
app.get('/', function (req, res) {
    console.log(uuidv1());
    res.sendFile(__dirname + '/index.html');
});
app.get('/id', function (req, res) {
    console.log(uuidv1());
    res.json({
        id: uuidv1()
    });
});
```

Battle Game - Multiplayer - index.html

```
<h1 class="message"></h1>
<div class="output"></div>
<input name="user" type="text" value="Laurence">
<button class="btn1">Join</button>
<button class="btn2">Roll</button>
<script src="/socket.io/socket.io.js"></script>
<script>
    const socket = io();
    const output = document.querySelector('.output');
    const message = document.querySelector('.message');
    const btn1 = document.querySelector('.btn1');
    const btn2 = document.querySelector('.btn2');
    const user = document.querySelector('input[name=user]');
    btn1.addEventListener('click', function () {
        let id = 'player_' + Math.floor(Date.now() * Math.random());
        socket.emit('new player', id, user.value);
        user.style.display = 'none';
        btn1.style.display = 'none';
    })
    btn2.addEventListener('click', function () {
        socket.emit('roll');
        btn2.disabled = true;
    })
```

```
socket.on('players', listPlayers);
    socket.on('inplay', checkWinner);

    function checkWinner(data) {
        message.innerHTML = data;
        btn2.disabled = false;
    }

    function listPlayers(players) {
        message.textContent = players.length > 0 ? `Round
${players[0].round} ` : `First Round 0`;
        output.innerHTML = 'Players<br>';
        players.forEach(function (player) {
            let div = document.createElement('div');
            div.textContent = `${player.name} roll = ${player.roll}`;
            div.style.color = player.winner ? 'green' : 'black';
            output.appendChild(div);
        })
    }
</script>
```

Battle Game - Multiplayer - app.js

```
const express = require('express');
const app = express();
const port = process.env.PORT || 3000;
const http = require('http').Server(app);
const io = require('socket.io')(http);
const lodash = require('lodash');
let players = [];
let round = 0;
const bodyParser = require('body-parser');
app.use(bodyParser.json())
app.use(bodyParser.urlencoded({
    extended: true
}));
app.use(express.static(__dirname + '/public'));
http.listen(port, function () {
    console.log("Ready " + port);
})
app.get('/', function (req, res) {
    res.sendfile("index.html");
});
app.get('/players', function (req, res) {
    res.send(players);
});
```

```
io.on('connection', function (socket) {
    let userId;
    socket.on('disconnect', function (reason) {
        players = players.filter(function (obj) {
            return obj.id !== userId;
        });
        nextRoundCheck();
    });
    socket.on('new player', function (id, name) {
        userId = id;
        players.push({
            name: name
            , id: id
            , round: round
            , roll: null
            , winner: false
        });
        io.emit('players', players);
    });
    socket.on('roll', function () {
        players.forEach(function (player) {
            if (player.id == userId) {
                player.roll = lodash.random(1, 1000);
            }
        })
        nextRoundCheck();
    });
    io.emit('players', players);
});
```

Battle Game - Multiplayer - app.js

```
function nextRoundCheck() {
    if (players.length > 0) {
        let ready = 0;
        let top = 0;
        let win = 0;
        players.forEach(function (player, index) {
            player.winner = false;
            if (player.roll) {
                ready++;
                if (player.roll && player.roll > top) {
                    win = index;
                    top = player.roll;
                }
            }
        })
        players[win].winner = true;
        io.emit('players', players);
        if (ready >= players.length) {
            io.emit('inplay', 'Round #' + round + ' winner is ' + players[win].name);
            round++;
            players.forEach(function (player, index) {
                player.winner = false;
                player.roll = null;
                player.round = round;
            })
        }
    }
}
```

Congratulations on completing the section

This ebook uses https://developer.mozilla.org/en-US/docs/Web/JavaScript as a source for examples. Check out more about JavaScript at MDN.

Find out more about my courses at http://www.discoveryvip.com/

Course instructor : Laurence Svekis - providing online training to over 500,000 students across hundreds of courses and many platforms.