Online Multiplayer Step By Step

1) download node.js. (http://nodejs.org/) You can build your backend in any of a million languages. node has a great package library and uses javascript (that I believe most of you know) and works well on heroku.

2) Download notepad++ or a similar text editor (sublime text 3 is popular these days)

3) open up the Node.js command prompt from the new nodejs folder. This will have all the node paths in the environment

4) let's create a folder for our project, and create server.js with some sample Hello World.

```
console.log( “Hello World” );
```

5) Create package.json file to describe your application. Each application in node is a package, and this tells the system hosting your app all the info it needs about your application. It also let's npm grab your dependencies automatically. You can create a package.json file easily by running the following command in your directory

```
npm init
```

Set your entry point to server.js when asked.

When done, open the package.json in your text editor and an engine and dependencies so the file looks more like the following:

Create a file called package.json with this in it:

```
{
  "name": "backend",
  "version": "0.0.1",
  "dependencies": {
          "express": "3.x"
  },
  "engine": {
          "node": "0.10.26",
          "npm": "1.3.x"
  }
}
```

Note the dependencies field. This is saying we required a package called express. Express provides all the services necessary to run an http server.

6) type npm install into the console window to install all dependencies needed by your app

7) now add this code to your server file. It will create an instance of an express app that you can then modify

```
var express = require('express');
var app = express();
var port = process.env.PORT || 5000;

app.listen(port, function() {
  console.log("Listening on " + port);
});
```

10) Now add a hello world end point so you can ask your browser to say HelloWorld. You'll have to restart your server to try it again. Use app.all so that you can reach the endpoint from GET or POST. Observe the callback convention in node:

```
app.all( "/helloWorld", function( req, res )
{
        res.end( "Hello World!" );
} );
```

11) let's build a leaderboard. We'll add an endpoint to set a score. Create setScore that takes a name and a score and displays that:

```
app.all( "/setScore", function( req, res )
{
        var name = req.query.name;
        var score = req.query.score;
        res.end( name + " got score " + score );
} );
```

12) let's get hosted now...Download and install the Heroku Toolbilt (https://toolbelt.heroku.com/) This will install ssh, git, and the heroku commands (built in Ruby) Make sure to install it into a path with NO SPACES.

13) create an account on heroku.com. It's free until you increase your dyno count.

14) quit the node cmd line and start using "git bash" in the git folder that was installed when you installed heroku. It's a linux style command line (so it stinks less than the dos one- also it has the ssh paths setup correctly). Try running your server again to make sure it works

15) Heroku uses git for deployment, so we need to create a git repository locally so we can push to heroku (if you want, you can create a git repository on github or bitbucket to push to as well, but that's outside the scope of this lecture)

16) login to heroku from the command line and enter your information. If you don't own the home directory on this computer, you'll want to move your ssh key from here when you're done (or keep it on a thumbstick) since it's your access to your app

17) download SourceTree from Atlassian. it's a great GUI git client. configure it with your details. you'll want to convert your ssh key to a putty key so you can push to heroku from sourcetree. follow instructions here (http://www.sourcetreeapp.com/faq/)

Tools > Create or Import SSH Keys
Conversions->Import key
Select your key generated by heroku (probably in %USERPROFILE%\.ssh\id_rsa)
Save the private and public keys in the same folder as PuttyPrivate and PuttyPublic

Then go to Options, Genera, SSH Client Configuration and browser to your key

18) in source tree, create a new repository in your folder, add all your files to the repository and commit them.

19) now create an app on heroku. Type the following lines:

```
git init
heroku apps:create name-of-your-app-that-is-unique
```

The first line will create a git repository and the second will add a remote to a new app on heroku

21) push your app to that remote in SourceTree. When asked if you want to start pageant, say yes. Show full output so you can see what's going on.

23) try hitting HelloWorld and setScore on your app. wooh, hosting! Free!

24) now it's time to add some kind of database, because we need to save those scores. Go to mongolab.com and sign up for an account. Create a free database.

25) create a username and password and copy the url to tell the driver how to connect

26) add the native mongo driver to your package.json: "mongodb": "1.3.20" and npm install

27) now let’s connect to our db! in general it’s better to put username and password in heroku config variables, so if you’re doing this on your own, look that up. Wrap all your code in the following: This will make sure our connection loads before we add the routes

```
var MongoClient = require('mongodb').MongoClient
MongoClient.connect(YOUR_INFO_HERE, function(err, db)
{
         if(err) throw err;

        //report any errors in callbacks
        db.on( 'error', function( err )
        {
                console.error( err );
                console.error( err.stack );
        } );

        //existing code goes here...

}
```

and in setScore, upsert the user with their score into the scores table…

```
//now let's add to the database!
var userCollection = db.collection( 'users' );
var query = { userName: name };
var update = { userName: name, score: score };
var options = { upsert: true };


userCollection.update( query, update, options, function( err, docs )
{
        if( err ) throw err;
        res.end( name + " got score: " + score );
}
```

If you start up supervisor again, you can test this locally without having to push to heroku. try setting a few scores. then go to your db on mongo labs and look at the contents

28) A Leaderboard is no good if you can’t fetch it. so let’s write an endpoint that fetches and sorts all the names.

```
app.all( "/getScores", function( req, res )
{
	var userCollection = db.collection( 'users' );
	userCollection.find( {}, { userName: true, score: true } ).
		sort( { score: -1 } ).
		toArray( function( err, items )
	{
		if( err ) throw err;

		res.end( JSON.stringify( items ) );
	} );
} );
```

29) now let’s access it from our code! See BackendFrontendDemo.zip file posted on blackboard