Contents

[Previous Class Topics 2](#_Toc45191985)

[Class Objectives 2](#_Toc45191986)

[Upcoming topics 3](#_Toc45191987)

[REPL 3](#_Toc45191988)

[REPL Commands 3](#_Toc45191989)

[ ctrl + c 3](#_Toc45191990)

[ ctrl + c twice 3](#_Toc45191991)

[ ctrl + d 3](#_Toc45191992)

[ Up/Down Keys 3](#_Toc45191993)

[ tab Keys 3](#_Toc45191994)

[ .help 3](#_Toc45191995)

[ .break 3](#_Toc45191996)

[ .clear 3](#_Toc45191997)

[ .save filename 3](#_Toc45191998)

[ .load filename 3](#_Toc45191999)

[var VS let VS const 4](#_Toc45192000)

[var: 4](#_Toc45192001)

[let: 4](#_Toc45192002)

[const: 4](#_Toc45192003)

[Variable Declaration Types 5](#_Toc45192004)

Class : Learning Node For Everyone

Start Date : 07/11/2020

Delayed class from July 4th holiday

## Previous Class Topics

* Introduction
  + What You Need
  + What is Node
  + Resources
  + Contacting Instructors
  + Class structure
* Getting Started
  + Installing
  + Setting up
    - Node
    - Mongo DB
    - MySQL
    - Editors (IDE)

## Class Objectives

* Developing Code (all languages)
  + Introduction to Concepts and Tools
  + Best Practices
* How node works
  + Your first server
    - Testing your first server
  + REPL Terminal
    - REPL commands
    - Using REPL
  + Online editor
    - Using online editor
* Your first homework
  + Starter code
  + Assignment

## Upcoming topics

* Async and Node
* DB with Node (Mongo & MySQL)
* Programming to handle errors
* Securing your code
* Modeling Data
* Using PUG
* Advanced Concepts

## REPL

REPL stands for Read Eval Print Loop and it represents a computer environment like a Windows console or Unix/Linux shell where a command is entered and the system responds with an output in an interactive mode. Node.js or Node comes bundled with a REPL environment.

* **Read** − Reads user's input, parses the input into JavaScript data-structure, and stores in memory.
* **Eval** − Takes and evaluates the data structure.
* **Print** − Prints the result.
* **Loop** − Loops the above command until the user presses **ctrl-c** twice.

## REPL Commands

* ctrl + c − terminate the current command.
* ctrl + c twice − terminate the Node REPL.
* ctrl + d − terminate the Node REPL.
* Up/Down Keys − see command history and modify previous commands.
* tab Keys − list of current commands.
* .help − list of all commands.
* .break − exit from multiline expression.
* .clear − exit from multiline expression.
* .save filename − save the current Node REPL session to a file.
* .load filename − load file content in current Node REPL sessi

Quick discussion about declarations.

## var VS let VS const

### var:

function scoped

undefined when accessing a variable before it's declared

### let:

block scoped

ReferenceError when accessing a variable before it's declared

### const:

block scoped

ReferenceError when accessing a variable before it's declared

can't be reassigned

To start your onw REPL terminal simply type NODE at the command line.

Try this. At the command line type node. Once the terminal loads. Type the following.

*x = 100*

*y = 1000*

*newx = 100*

*x + newx*

*console.log("I am your message to the user of this command prompt")*

Now lets looks at multi line input. Type the following in a text editor.

*do {*

*variabledoneright++;*

*console.log("variabledoneright: " + variabledoneright);*

*}*

*while ( variabledoneright < 5 );*

Now at the NODE command in REPL past this code.

Observe the output. Now type the following.

Variabledoneright = 1

Then paste the code again and observe the outcome.

## Variable Declaration Types

Type the following into a text editor. Then observe the outcome.

let variablechaning = 'Kim Kardashian'

variablechaning = 'Kim Kardashian West'

const variableunchanging = {

name: 'Kim Kardashian'

}

variableunchanging.name = 'Kim Kardashian West'

variableunchanging = {}

Now observe the outcome. Const variables cannot be reassigned but let variables can be.

Open the browser based REPL and let’s do some more code.

<https://www.tutorialspoint.com/execute_nodejs_online.php>

Let’s put the following code in the online tester.

*function discountPrices (prices, discount) {*

*let discounted = []*

*for (let i = 0; i < prices.length; i++) {*

*let discountedPrice = prices[i] \* (1 - discount)*

*let finalPrice = Math.round(discountedPrice \* 100) / 100*

*discounted.push(finalPrice)*

*}*

*console.log(i) // 3*

*console.log(discountedPrice) // 150*

*console.log(finalPrice) // 150*

*return discounted*

*}*

*console.log(discountPrices([100, 200, 300], .5));*

Notice the error. Now lets remove the following lines.

*console.log(i) // 3*

*console.log(discountedPrice) // 150*

*console.log(finalPrice) // 150*

Now observe the outcome.

For this class we will be slowly building a simple application. The first step is to output a grid on the screen that will become a simple maze later for our game.

The following is our starter code.

Create a file called PayaMazer.js with the following code.

*'use strict';*

*/\*\**

*\* Generates the maze asynchronously.*

*\* @param {Number} x Width of the maze.*

*\* @param {Number} y Height of the maze.*

*\* @returns {Promise} finished when resolved.*

*\*/*

*function maze(x,y) {*

*return new Promise((resolve, reject) => {*

*let n=x\*y-1;*

*if (n<0) {*

*reject(new Error(`illegal maze dimensions (${x} x ${y} < 1)`));*

*} else {*

*let horiz =[]; for (let j= 0; j<x+1; j++) horiz[j]= [];*

*let verti =[]; for (let j= 0; j<x+1; j++) verti[j]= [];*

*let here = [Math.floor(Math.random()\*x), Math.floor(Math.random()\*y)];*

*let path = [here];*

*let unvisited = [];*

*for (let j = 0; j<x+2; j++) {*

*unvisited[j] = [];*

*for (let k= 0; k<y+1; k++)*

*unvisited[j].push(j>0 && j<x+1 && k>0 && (j != here[0]+1 || k != here[1]+1));*

*}*

*while (0<n) {*

*let potential = [[here[0]+1, here[1]], [here[0],here[1]+1],*

*[here[0]-1, here[1]], [here[0],here[1]-1]];*

*let neighbors = [];*

*for (let j = 0; j < 4; j++)*

*if (unvisited[potential[j][0]+1][potential[j][1]+1])*

*neighbors.push(potential[j]);*

*if (neighbors.length) {*

*n = n-1;*

*let next= neighbors[Math.floor(Math.random()\*neighbors.length)];*

*unvisited[next[0]+1][next[1]+1]= false;*

*if (next[0] == here[0])*

*horiz[next[0]][(next[1]+here[1]-1)/2]= true;*

*else*

*verti[(next[0]+here[0]-1)/2][next[1]]= true;*

*path.push(here = next);*

*} else*

*here = path.pop();*

*}*

*resolve({x: x, y: y, horiz: horiz, verti: verti});*

*}*

*});*

*}*

*/\*\**

*\* A mere way of generating text.*

*\* The text (Since it can be large) is generated in a non-blocking way.*

*\* @param {Object} m Maze object.*

*\* @param {Stream} writeTo Optinally, include here a function to write to.*

*\* @returns {Promise} finished when the text is generated.*

*\*/*

*function display(m, writeTo) {*

*return new Promise((resolve, reject) => {*

*let text = [];*

*for (let j= 0; j<m.x\*2+1; j++) {*

*let line = [];*

*if (0 == j%2)*

*for (let k=0; k<m.y\*4+1; k++)*

*if (0 == k%4)*

*line[k] = '+';*

*else*

*if (j>0 && m.verti[j/2-1][Math.floor(k/4)])*

*line[k] = ' ';*

*else*

*line[k] = '-';*

*else*

*for (let k=0; k<m.y\*4+1; k++)*

*if (0 == k%4)*

*if (k>0 && m.horiz[(j-1)/2][k/4-1])*

*line[k] = ' ';*

*else*

*line[k] = '|';*

*else*

*line[k] = ' ';*

*if (0 == j) line[1] = line[2] = line[3] = ' ';*

*if (m.x\*2-1 == j) line[4\*m.y]= ' ';*

*text.push(line.join('')+'\r\n');*

*}*

*const OUTPUT = text.join('');*

*if (typeof writeTo === 'function')*

*writeTo(OUTPUT);*

*resolve(OUTPUT);*

*});*

*}*

*module.exports = {*

*maze: maze,*

*display: display*

*}*

Now create a file called PayaMazerMain.js with the following code.

*'use strict';*

*const maze = require('./PayaMazer.js');*

*const X = 20,*

*Y = 20;*

*console.log(`Generating a maze of ${X} x ${Y}...`);*

*const origin = new Date().getTime();*

*maze.maze(X, Y).then((m) => {*

*const time = new Date().getTime() - origin;*

*console.log(`Done in ${time <= 1000 ? time+'ms' : Math.round(time/1000)+'s'}!`);*

*maze.display(m, console.log);*

*//Here you can pass a given stream (ie: stream) and it's write function;*

*//An example could be: maze.display(m, stream.write);*

*}, (err) => console.error(err));*

Your assignment is to test this code by making java script files on your disk and executing from node. Also see if you can figure out how to test the functions in the online generator and from REPL. Comment the code with what it is doing and why to the best of your ability. In other words adding to my comments where it helps you understand it more.