* <!DOCTYPE html> and the <html> tag tell you that the language of the file is written in HTML which is a hypertext markup language specifically in this case HTML 5
* Html tag also contains everything that is in the web page and tells the browser that this is an HTML document.
  + The lang attribute inside the html tag tells you what language you are using for this html file which in this case is in english
* The <head> tag holds important information about the format of the html file that users may or wont see at all. It holds the <title> tag which display the title of the web page about the browser or where the new tab will be.
* The meta tag/element found inside the head tag provides structured **metadata** about a Web page, which are data that provides info about other data
* The head tag also holds the <link> tag holds important links to other important external resource and information such as the CSS stylesheet which dictates how the page would look when user loads the page.
* Href attribute would be where the file can be found in order to link the two documents together.
  + It can specifies the URL of the page the link goes to
* The rel attribute specifies the relationship between the current document and the linked document. Only used if the **href attribute** is present
* <body> tag holds/contains all the information that will be displayed on the web page when user loads the page
* <script> tag holds the type of script or declare the type of script that’ll be used within the html in this case it uses JavaScript, hence the text attribute and the src attribute links where the source of where the JavaScript is found.
* UTF-8
  + Is a Unicode that are used to save the format of the file which tells the browser

Explain the box model, its four component and related CSS properties

* The box model consist of 4 components which is the
  + Content which holds the content/material that the users will see
  + Padding which is the invisible space between your content and the border of of the box containing everything meaning padding and your content
  + Border is what contains the content and padding. It is also visible to the User when we choose to display it in CSS
    - Box-sizing: border-box; allows CSS to now include the calculation of the full width of the entire container/box
  + Margins are the invisible space between boxes
    - It bascally space your box how much you want it relative to its surrounding objects.
    - When having two object with two different margins the larger margins takes over and does not combine the two margins space between it.

What is a self-closing tag \

17.  What is the value of myValue after running the code below? var myValue = 0; var myArray = [1, 2, 3, 4]; for (var i = 1; i < myArray.length; i++) { myValue += myArray[i]; }

~~10~~

9

19.  We can get the key a user has pressed by referencing \_\_\_\_.

~~keyboard.event~~

event.key

22. Repetitive lines of code should be bundled together as \_\_\_\_ and then executed as needed.

~~arrays~~

functions

24. To grab the first letter of a string we can use the \_\_\_\_ method.

~~stringItem(0)~~

charAt(0)

25.  What will the following code log to the console? var myArray = []; for (var i = 0; i < 10; i++) { if (i % 2 == 0) { myArray.push(i); } } console.log(myArray);

~~[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]~~

[0, 2, 4, 6, 8]

29. What will be logged to the console with the following code: var value = 33; function myFunction() { value = 66; } myFunction(); console.log(value);

~~33~~

66

* setTimeout
  + when need something to happened at a specify time
  + takes 2 argument
    - callback
    - time
* setInterval
  + setting the amount on the number between the increment/decrement
* clearTimeout
  + clears the timeout and cancels it
  + takes 1 argument
    - the variable that clears it

Top of Form

More JavaScript and jQuery Checkpoint

Page 1 Questions

Hey Phi Hai Nguyen,  
  
Please take a few minutes to complete this Checkpoint.

1. Without using jQuery, a CSS style of an element can be manipulated using the \_\_\_\_ method.

* setStyle
* setAttribute
* adjustCSS
* .css()

2. To attach an element to the beginning of another element, we can use the \_\_\_\_ jQuery method.

* .attach
* .prepend
* .append
* .start

3.  In terms of objects, how would one use bracket notation to refer to a value that is not a variable?

* object(property)
* object["property"]
* object("property")
* object-property

4. What is another JavaScript method that is used to loop over arrays or Objects?

* readAll
* $.each
* loopOver
* forEach

5. A div with the id of myDiv can be targeted using jQuery by typing:

* $("#myDiv")
* $div
* $("myDiv")
* $(.myDiv)

6. Without using jQuery, we can code a new div with the following:

* createNew("div")
* $("<div>")
* createElement("div")
* makeDivElement()

7. Once a new div has been coded, it can be attached to an element on the page with the \_\_\_\_ JavaScript method.

* appendChild
* createPosition
* setElement
* placeDiv

8. In terms of objects, how would one use dot notation to refer to a value?

* property.object
* object.[property]
* object.property
* property.[object]

9. The \_\_\_\_ jQuery method can be used to iterate over an array without using a for loop.

* .iterate
* .each
* .every
* .loop

10. Variables that we can refer to everywhere are known as \_\_\_\_ variables.

* primary
* permanent
* global
* eternal

11. A new button can be created using jQuery and assigned to a variable using with the following code:

* var newButton = $("<button>");
* var newButton = $newButton;
* var newButton = create.button;
* var newButton = button("new");

12.  We can use jQuery to create a listener that listens for a click on all elements with the class myClass using:

* $(".myClass").on(clickMe!, function() { ... });
* $(".myClass").on("click", function() { ... })
* $(".myClass").on("press", function() { ... });
* $(".myClass").on(function() { ... });

13. When defining an object, its property-value pairs are enclosed within \_\_\_\_.

* curly brackets
* colons
* parentheses
* square brackets

14.  Effectively, the \_\_\_\_ is the version of HTML / CSS interpreted by the browser.

* debugger
* DOM
* program
* console

15. Without using jQuery, elements on a page can be populated using the \_\_\_\_ property.

* script
* div
* innerHTML
* .text()

16.  A button with an id of myButton can be set to have its displayed name show as "SuperButton" with the following jQuery code:

* $("#myButton").set("SuperButton");
* $("#myButton").text("SuperButton");
* $#myButton("SuperButton");
* $("#myButton").name("SuperButton");

17. What will be logged to the console with the following code?

var myFunction = function(someCar) {

var myCar = someCar;

}

myFunction("Honda");

console.log(myCar);

* Honda
* someCar
* myFunction
* ReferenceError: myCar is not defined

18. With the code below, how would you log  "lizard" to the console?

var group = {

large: {

creatures: "blue whale",

medium: {

creatures: ["zebra", "rhino", "hippo"],

small: {

creatures: "cat",

tiny: {

creatures: ["snail", "hamster", "lizard", "spider"]

}

}

},

other: {

creatures: "human"

}

}

};

* console.log(group.tiny.creatures[2]);
* console.log(group.large.medium.small.tiny.creatures[2]);
* console.log(group.tiny.creatures.lizard);
* console.log(group.large.medium.small.tiny.creatures[lizard]);

19. A page contains a div with the id "emptyDiv".  jQuery may be used to attach a div named newDiv to it with the following code:

* ("emptyDiv").appendChild(newDiv);
* $(emptyDiv).appendChild(newDiv);
* $("#emptyDiv").append(newDiv);
* $("#emptyDiv").attach(newDiv);

20.  \_\_\_\_ makes sure that our JavaScript code doesn't get run until the HTML document is finished loading.

* page.loaded
* document.ready
* document.complete
* process.ended

21.  jQuery can be useful for tasks such as:

* Dynamically inserting, updating, or removing HTML
* Registering click or other change events
* Downloading data from databases
* All of the above

22. Functions that are properties of objects are also known as \_\_\_\_.

* tasks
* methods
* steps
* functions

23. jQuery is a cross-platform JavaScript \_\_\_\_ for easier client-side scripting.

* library
* command
* table
* template

24.  The \_\_\_\_\_ jQuery method can be used to place attributes on HTML elements

* .info
* .attribute
* .append
* .attr

25. What will the code below log to the console when run in a browser?

function myFunction() {

console.log(this);

}

myFunction();

* this
* null
* Window
* myFunction

26. We can give classes to elements using the \_\_\_\_ jQuery method.

* .attrClass
* .giveClass
* .addClass
* .setClass

27.  What will be logged to the console with the following code?

var myObject = {

myNumber: 30,

myFunction: function() {

this.myNumber += 30;

console.log(myObject.myNumber);

}

}

myObject.myFunction();

* undefined
* 3030
* 30
* 60

28. The \_\_\_\_ keyword is simply a way to reference the object itself.

* this
* item
* object
* it

29. Writing \_\_\_\_ is equivalent to writing $() when using the jQuery library.

* library()
* jQuery()
* select()
* click()

30.  In JavaScript, \_\_\_\_ functions can access \_\_\_\_ variables, but not vice versa.

* parent, child
* all, child
* child, parent
* parent, all

0%

Top of Form

Timers and APIs Checkpoint

Hey Phi Hai Nguyen,

1.  What JavaScript method runs a function after a specified amount of time?

* setTimeout

2. The \_\_\_\_ JavaScript method can be used to run a function every specified amount of time passes.

* setInterval

3. To end a timer that was previously set, we can use the \_\_\_\_ JavaScript method.

* clearTimeout

4. What will be the result of running the following code:

var windowTimeout = setInterval(function() {

console.log("hi!");

}, 2000);

* The word "hi!" will be logged to the console every two seconds every two seconds

5. The functions passed to setTimeout and setInterval are examples of \_\_\_\_ functions.

* inner
* standard
* outer
* callback

6. A set of steps that a computer can take to solve a particular problem is known as \_\_\_\_.

* an algorithm

7. API is an acronym for \_\_\_.

* Application Programming Interface

8. AJAX is an acronym for \_\_\_.

* Asynchronous JavaScript and XML

9. What two parameters must we pass into AJAX to retrieve data from online?

* url and method

10. JSON is an acronym for \_\_\_.

* JavaScript Object Notation

11. APIs can be used for:

* retrieving data from someone else's database
* utilizing someone else's more complex functionality
* controlling other hardware and software
* all of the above

12. The total count of APIs in existence is \_\_\_.

* over 1,000,000

13. \_\_\_\_ is the common format for sending data between APIs.

* JSON

14. In the URL, \_\_\_ can be included in order to filter the results from an API.

* query parameters

15.  What is the best resource to properly use a specific API?

* The API's official documentation

16. In the code below, where is all the data from the API?

$.ajax({

url: queryURL;

method: "GET"

}).done(function(res) {

console.log("completed");

});

* within the res variable

17.  In the code below, what will be logged to the console?

$.ajax({

url: queryURL;

method: "GET"

}).done(function(res) {

console.log("A");

});

console.log("B");

* B then A

18. JavaScript is \_\_\_ by nature.

* synchronous

19. Say that we execute process A and then continue with the execution of other processes while process A is still executing. Process A is then resolved after some time passes and the callback function within process A executes the code.  
  
Process A is an example of \_\_\_\_.

* asynchronous programming

0%

Bottom of Form

// Given a non-empty array of integers, every element appears twice except for one. Find that single one.

// Example

// Input: [2, 2, 1]

// Output: 1

function singleNumber(arr){

console.log(arr)

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

var currentNum = arr[i]

arr[i]=null//output => [nul, 1, 2]

if (!arr.includes(currentNum)){

return currentNum;

}

arr[i]=currentNum;

Bottom of Form

// Write a function that takes in an array of integers and returns an array with the duplicates removed

// Ex:

// Input: [1,2,2,3]

// Output: [1,2,3]

// Input: [4,5,4,4,7,5]

// Output: [4,5,7]

// Input: [1,2,3,5]

// Output: [1,2,3,5]

function singleArr(arr){

for(var i =0; i < arr.length; i++) {

for (var j =i+1; j<arr.length; j++ ){

if(arr[i] === arr[j]) {

arr.splice(i, 1);

}

}

}console.log(arr)} singleArr([1,2,2,3,2,5,3,6,2])

//===========this is working below=====================//

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

const element = array[i];

if (newArray.indexOf(arr[i]=== -1){

newArray.push(arr[i])

} return newArray;}

NODE BEGINNER

\* Create a command-line node application that takes in parameters like this:

\* `node calculator.js add 1 2` ... and outputs 3

\* `node calculator.js subtract 5 2` ... and outputs 3

\* `node calculator.js multiply 3 2` ... and outputs 6

\* `node calculator.js divide 8 2` ... and outputs 4

\* `node calculator.js remainder 7 2`... and outputs 1

**### Bonuses**

\* Enable your calculator application to also handle the below cases:

\* `node calculator.js exp 7 2` ... and output 49 (7 squared)

\* `node calculator.js algebra 4x+2=10`... and output 2. (Hint: Assume the algebra will always be in this exact form and will always be addition)

console.log(parseFloat(process.argv[2]), parseFloat(process.argv[3]));

var input = process.argv;

var operand = input[2];

var num1 = input[3];

var num2 = input[4]

var results;

if (operand === "add"){

results = parseFloat(num1) + parseFloat(num2)

console.log(results)

}else if(operand === "minus"){

results = parseFloat(num1) - parseFloat(num2)

}else if(operand === "multiply"){

results = parseFloat(num1) \* parseFloat(num2)

}else if(operand === "divide"){

results = parseFloat(num1) / parseFloat(num2)

}else if(operand == "remainder"){

results = parseFloat(num1) % parseFloat(num2)

}

console.log(results)

// Write a function that takes a string as a parameter and determines if it is a valid password with the following constraints:

// 1. It must be at least 8 characters long

// 2. It must contain at least 1 capital letter

// 3. It must contain at least 1 lower case letter

// Output true if the string is a valid password or false otherwise.

// Ex:

// Input: RexTheDog

// Output: true

// Input: rexthedog

// Output: false

// Input: REXTHEDOG

// Output: false

// Input: Dog

// Output: false

function password(str){

var upperCase = false;

var lowerCase = false;

if(str.length<8){

return false;

}

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

if (str[i]==str[i].toUpperCase()){

upperCase=true;

}else if(str[i]==str[i].toLowerCase()){

lowerCase=true;

}

if(lowerCase && upperCase){

return true;

}

}

return false;

}console.log(password('REXTHEDOG'))

// A pangram is a sentence that contains all the letters of the English alphabet at least once.

// \* Write a function that takes a string and determines whether the string is a pangram.

// BONUS: \*\*without\*\* typing out the full alphabet anywhere in your code.

// For example: `The quick brown fox jumps over the lazy dog.`

// OUTPUT => true

var isPangram = function (str) {

if (str.length < 26) {

return false;

}

var newArray = str.split(" ").join("").toLowerCase().split("");

//first split => ['The', 'quick']

//after the join => ['Thequick']

//after toLowerCase => ['thequick']

// after split again => ['t', 'h', 'e'...]

var letterHolder = [];

//loop over newArray

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

// if newArray[i] is not in letterHolder, push the value into our letterHolder

if (!letterHolder.includes(newArray[i])) {

letterHolder.push(newArray[i])

}

}

//if new array equal 26, return true, stry is pangram

if (letterHolder.length === 26) {

return true;

}else {

return false;

}

}

// Given an integer, write a function to determine if it is a power of three.

// Example 1:

// Input: 27

// Output: true

// Example 2:

// Input: 0

// Output: false

// Example 3:

var powerofTHree = function(N){

//check to see if 3 to the xth power equal N

// if true, return true. N is a power of 3

//else if the 3 to the xth power is greater than N, then we know we gone too far, so we know that N is not a power of 3

//increment x by 1 and run iteration again

for (var i =0; ; i++) {

if (Math.pow(3, i)===N) {

return true

}

else if (Math.pow(3, i)>N) {

return false;

}

}

}

powerofTHree(27);

RPG

// constructor function which can take in a series of values and create objects

// with the properties contained inside

function Character(name, profession, gender, age, strength, hitpoints) {

this.name = name;

this.profession = profession;

this.gender = gender;

this.age = age;

this.strength = strength;

this.hitpoints = hitpoints;

// method which prints all of the stats for a character

this.printStats = function() {

console.log("Name: " + this.name + "\nProfession: " + this.profession +

"\nGender: " + this.gender + "\nAge: " + this.age + "\nStrength: " +

this.strength + "\nHitPoints: " + this.hitpoints);

console.log("\n-------------\n");

};

// method which determines whether or not a character's "hitpoints" are less than zero

// and returns true or false depending upon the outcome

this.isAlive = function() {

if (this.hitpoints > 0) {

console.log(this.name + " is still alive!");

console.log("\n-------------\n");

return true;

}

console.log(this.name + " has died!");

return false;

};

// method which takes in a second object and decreases their "hitpoints" by this character's strength

this.attack = function(character2) {

character2.hitpoints -= this.strength;

};

// method which increases this character's stats when called

this.levelUp = function() {

this.age += 1;

this.strength += 5;

this.hitpoints += 25;

};

}

// creates two unique characters using the "character" constructor

var warrior = new Character("Crusher", "Warrior", "Male", 25, 10, 75);

var rogue = new Character("Dodger", "Rogue", "Female", 23, 20, 50);

warrior.printStats();

rogue.printStats();

rogue.attack(warrior);

warrior.printStats();

warrior.isAlive();

rogue.levelUp();

rogue.printStats();

// while loop that continues to run so long as both characters' "hitpoints" are above zero

while (warrior.isAlive() === true && rogue.isAlive() === true) {

// characters deal damage to one another

warrior.attack(rogue);

rogue.attack(warrior);

// prints stats to show changes

warrior.printStats();

rogue.printStats();

INQUIRER

// dependency for inquirer npm package

var inquirer = require("inquirer");

// constructor function used to create programmers objects

function Programmer(name, position, age, language) {

this.name = name;

this.position = position;

this.age = age;

this.language = language;

}

// var arron = new Programmer("Arron", "Instructor", 27, "JS")

// var jean = new Programmer("Jean", "Programmer", 29, "JS")

// jean.printStats();

// arron.printStats();

Programmer.prototype.printInfo = function(){

console.log(this);

}

// runs inquirer and asks the user a series of questions whose replies are

// stored within the variable answers inside of the .then statement

inquirer.prompt([

{

name: "name",

message: "What is your name?"

}, {

name: "position",

message: "What is your current position?"

}, {

name: "age",

message: "How old are you?"

}, {

name: "language",

message: "What is your favorite programming language?"

}

]).then(function(answers) {

// initializes the variable newProgrammer to be a programmer object which will take

// in all of the user's answers to the questions above

var newProgrammer = new Programmer(answers.name, answers.position, answers.age, answers.language);

// printInfo method is run to show that the newProgrammer object was successfully created and filled

newProgrammer.printInfo();

});

// Maximin

// Problem

// Write a function that finds the minimum and maximum elements from an array without using the Math.min or Math.max methods.

// Example

// Input => [1, 2, 3, 4, 5]

// Ouput => [1, 5]

function maxmin(arr){

//declare var that sets the min and max

var min = arr[0];

var max = arr[0];

//loop around arr

for (var i = 0; i < arr.length; i++) {

if ( arr[i]<min){

min = arr[i];

}

if(arr[i]>max){

max=arr[i];

}

return [min, max]

}

}maxmin([5, 1, 4, 2, 3]);

// Given a List of words, return the words that can be typed using letters of alphabet on only one row of the American keyboard like the image below.

// 1 2 3 4 5 6 7 8 9 0 - =

// Q W E R T Y U I O P [ ] \

// A S D F G H J K L ; '

// Z X C V B N M , . /

// Example:

// Input: ["Hello", "Alaska", "Dad", "Peace"]

// Output: ["Alaska", "Dad"]

WHITEBOARDING

* function that takes in an array of numbers and returns the sum
  + - var sum = 0;
  + for(var i=0; i<numbers.length; i++){
    - sum+=numbers[i]}
    - return sum
* function that takes in a number as age and see if person is old enough to drink
* function that takes in array of number and output the avg of all the numbers
* function that takes in a number and returns the factorial of that number without using a for loop

MYSQL EXAMPLE

// Table: Person

// +-------------+---------+

// | Column Name | Type |

// +-------------+---------+

// | PersonId | int |

// | FirstName | varchar |

// | LastName | varchar |

// +-------------+---------+

// PersonId is the primary key column for this table.

// Table: Address

// +-------------+---------+

// | Column Name | Type |

// +-------------+---------+

// | AddressId | int |

// | PersonId | int |

// | City | varchar |

// | State | varchar |

// +-------------+---------+

// AddressId is the primary key column for this table.

// Write a SQL query for a report that provides the following information for each person in the Person table, regardless if there is an address for each of those people: FirstName, LastName, City, State

// What type of join will we need? Why?

You will need a left join to join everything to the left table even if there are no matching value

// SQL Query

SELECT person.firstName, person.lastName, address.city, address.state FROM person LEFT JOIN address on person.personId = address.personId ORDER BY personId desc;

Top of Form

# Node Checkpoint

## quiz

Hey Phi Hai Nguyen,  
  
Please take a few minutes to complete this Checkpoint.

1. \_\_\_ allows us to quickly incorporate pre-made code snippets into Node applications.

* An API
* The Node Package Manager
* GitHub
* Web scraping

2. The internal Node package fs is used for \_\_\_.

* creating NPM packages
* making AJAX requests
* DOM updates
* interacting with files

3. The contents of the file data.txt are i,love,coding. What will be the output of the code below?

var fs = require("fs");

fs.readFile("data.txt", "utf8", function(err, data) {

var output = data.split(",");

console.log(output);

});

* i love coding
* i,love,coding
* {data: "i, love, coding"}
* ['i', 'love', 'coding']

4. The \_\_\_ is the physical hardware or software that takes requests from the \_\_\_ and gives something back or completes a process.

* client, API
* API, client
* client, server
* server, client

5. The file foods.js contains the code shown below. How would snickers be referenced in another file that had imported foods.js into the variable foods?

var yummies = {

candy: "snickers",

soda: "pepsi",

chips: "lays"

}

module.exports = {

yummies: yummies

}

* foods.yummies
* foods.yummies.candy
* foods.candy
* yummies.candy

6. The parseFloat and parseInt JavaScript functions can be used to parse an argument into \_\_\_.

* an array
* a string
* a number
* a query

7. Asking for something or some process to happen is called the \_\_\_ while what is returned as a result of it is called the \_\_\_.

* AJAX, web page
* request, API
* response, request
* request, response

8.  The \_\_\_ method of the Node fs package is used to add content to an existing file.

* appendFile
* writeFile
* appendToFile
* continueFile

9. The code below shows an example of using the request NPM package to make a call to an API. Assume no API key is required. The useful data will be contained in the \_\_\_ variable as \_\_\_.

var request = require("request");

request("http://www.omdbapi.com/?t=avatar", function(err, response, body) {

*// do something*

});

* err, an integer
* response, an array
* body, a string
* body, a JSON object

10. When writing JavaScript files that use Node, we can bring in code exported from another file using the \_\_\_ keyword.

* require
* imports
* module.imports
* require.imports

11.  The JSON.parse() method parses a JSON \_\_\_, constructing the JavaScript object it describes.

* module
* object
* array
* string

12.  The advantage of using Node as a server is:

* Easy extendability with plugins
* JavaScript use so frontend JavaScript devs can code backend too
* Fast implementation with few lines of code
* all of the above

13. The process.argv property can be used to get all the \_\_\_.

* files in the folder
* command line arguments
* active Node threads
* bash/terminal process history

14. \_\_\_ allows us to run JavaScript outside of the browser, and on the server instead.

* HTTP
* NPM
* AJAX
* Node

15. When a Node application with correctly listed dependencies is shared, the subsequent developer can run the command \_\_\_ to re-download all the required NPM packages.

* npm install
* node add dependencies
* node install -all
* npm add -all

16. The file runMe.js contains the code shown below. What will be the output of executing the command  node runMe.js 5 5?

console.log(process.argv[2] + process.argv[3]);

* 23
* 55
* 5
* 10

17. We can use Node to run a JavaScript file outside of the browser from bash/terminal with the command \_\_\_.

* You cannot - it must be run from the browser
* run filename.js
* node filename.js
* filename.js execute

18. Any NPM packages that an application is dependent on should be listed in \_\_\_.

* a separate text file
* the package.json file
* a readme file
* the GitHub repository

19. The file hello.js includes the code shown below. Where would the output be seen if hello.js was run using node from a bash/terminal window?

console.log('Hello')

* no output would be seen
* the browser's developer tools console
* the browser window
* the bash/terminal window

20. Node uses \_\_\_ threading which allows the server to handle all requests using a single thread through event-based callbacks.

* synchronous
* asynchronous
* multi
* perpendicular

21. External packages needed by your Node app can be added with the command:

* node add <packagename>
* npm add <packagename>
* node install <packagename>
* npm install <packagename>

22. What command is used to create a package.json file?

* npm add package.json
* node package.json --save
* npm init
* node create --add --save

0%

Bottom of Form

Top of Form

# Node Checkpoint

## Quiz Score

Your Results:

You passed! Your score was 68%

4. The \_\_\_ is the physical hardware or software that takes requests from the \_\_\_ and gives something back or completes a process.

~~client, server~~

server, client

20. Node uses \_\_\_ threading which allows the server to handle all requests using a single thread through event-based callbacks.

~~synchronous~~

asynchronous

16. The file runMe.js contains the code shown below. What will be the output of executing the command  node runMe.js 5 5? console.log(process.argv[2] + process.argv[3]);

~~10~~

55

5. The file foods.js contains the code shown below. How would snickers be referenced in another file that had imported foods.js into the variable foods? var yummies = { candy: "snickers", soda: "pepsi", chips: "lays" } module.exports = { yummies: yummies }

~~yummies.candy~~

foods.yummies.candy

8.  The \_\_\_ method of the Node fs package is used to add content to an existing file.

~~appendToFile~~

appendFile

9. The code below shows an example of using the request NPM package to make a call to an API. Assume no API key is required. The useful data will be contained in the \_\_\_ variable as \_\_\_. var request = require("request"); request("http://www.omdbapi.com/?t=avatar", function(err, response, body) { // do something });

~~body, a JSON object~~

body, a string

11.  The JSON.parse() method parses a JSON \_\_\_, constructing the JavaScript object it describes.

~~object~~

string

100%

Bottom of Form

// A self-dividing number is a number that is divisible by every digit it contains.

// For example, 128 is a self-dividing number because 128 % 1 == 0, 128 % 2 == 0, and 128 % 8 == 0.

// Also, a self-dividing number is not allowed to contain the digit zero.

// Given a lower and upper number bound, output a list of every possible self dividing number, including the bounds if possible.

// Example 1:

// Input:

// left = 1, right = 22

// Output: [1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 15, 22]

//

function selfDividing(left, right){

//store all numbers bewteen left and right including left and right in an array

var output = [];

if(left<10){

for (let i = 0; i < 10; i++) {

output.push(1);

}

for (let i = 10; i < right+1; i++) {

var selfDiv = true;

//becasue i is an integer we must turn it into a string to iterate through the current number

var currentNum = i.toString();

for (let k = 0; k < currentNum.length; k++) {

//parseInt is used to turn the individual number in the current number and seeing if it returns the number as selfdividing

if (i % parseInt(currentNum[k]) !==0) {

selfDiv = false;

}

}

if(selfDiv){

output.push(i)

} }

//this next code will help check for integers greater than 10

}else{

for (let i = left; i < right + 1; i++) {

var selfDiv=true;

}

}

// Given an array A of non-negative integers, return an array consisting of all the even elements of A, followed by all the odd elements of A.

// You may return any answer array that satisfies this condition.

// Example 1:

// Input: [3,1,2,4]

// Output: [2,4,3,1]

// The outputs [4,2,3,1], [2,4,1,3], and [4,2,1,3] would also be accepted.

function sortArray(arr){

var evenArr=[];

var oddArr=[];

var newArr=[];

// iterate around arr

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

// determine if the index i is an even or oddnumber

if (arr[i]%2===0) {

evenArr.push(arr{[i])

}else{

oddArr.push(arr[i])

}

}

newArr=evenArr.concat(oddArr);

return newArr

or

//the bottom will concatenate the arrays by sing a ES6 method

return […evenArr, …oddArr];

// if the current index is divisiable by 2 it will return a remainder of 0

// if even push into evenArr

//else push into oddArr

//concatenate our arrays with even as our left array

// return new array

};

console.log(sortArray([3,1,2,4]))

Top of Form

# MySQL Checkpoint

Hey Phi Hai Nguyen,  
  
Please take a few minutes to complete this Checkpoint.

1. Each record in a table is uniquely identified by a \_\_\_ key.

* master
* singular
* primary
* unique

2. An SQL statement is executed that successfully adds a new record of a person's name and age to a table. What is the result of executing the exact same SQL statement a second time?

* The second time code execution will be ignored
* A new identical record will be created
* The original record will be overwritten
* This will cause a database error

3. With very large databases, using certain columns as \_\_\_ helps speed up evaluation of queries.

* indexes
* primary keys
* null space
* foreign keys

4. When modifying existing table data, the \_\_\_ keyword is used to specify the column(s) and new value(s).

* WHERE
* SET
* FIELD
* LOCATION

5. Existing records in a table can be modified with the \_\_\_ statement.

* UPDATE
* RESET
* REDO
* MODIFY

6. The \_\_\_ constraint is used to add a specific value to a column when no other value has been specified.

* DEFAULT
* SELECTED
* USE
* VALUE

7. MySQL developers use \_\_\_ files to save their database and table creation code, and \_\_\_ files to save the statements they use to insert data into their tables.

* seeds.sql, schema.sql
* database.sql, data.sql
* schema.sql, seeds.sql
* data.sql, database.sql

8. Columns can be prevented from having empty fields for any record by defining them with the \_\_\_ constraint.

* REQUIRE DATA
* NOT NULL
* MUST VARCHAR
* PREVENT EMPTY

9. \_\_\_ can be utilized to combine two or more individual tables together using a value that is shared between them.

* Joins
* Combines
* Outers
* Inners

10. The correct syntax for making a new table called toys is \_\_\_.

* MAKE TABLE NEW toys ();
* INIT toys TABLE ();
* CREATE TABLE toys ();
* START toys TABLE ();

11.  Data may be removed from a specified table by using the \_\_\_ statement.

* DELETE RECORD
* DELETE FROM
* REMOVE DATA
* REMOVE FROM

12.  The \_\_\_ data type can be used to specify that a column may only hold whole numbers.

* VARCHAR
* NUMBER
* BLOB
* INTEGER

13.  \_\_\_ generates a new value for each inserted record in a table, increasing it by 1 each time by default.

* AUTO\_INCREMENT
* DEFAULT\_ADD
* UPDATE\_ID
* ADD\_INTEGER

14.  MySQL is to \_\_\_, as MySQL Workbench and Sequel Pro are to \_\_\_.

* JavaScript, SQL
* server, client
* Windows, Mac
* Node, JavaScript

15. What does SQL stand for?

* Stored Query List
* Structured Query Language
* Stripe Quality Lambda
* Strict Qualified Language

16. A database must first be \_\_\_ before attempting to connect to it with Node.

* emptied
* deleted
* populated
* created

17. A new database called friends\_db can be created using the following SQL code:

* MAKE DATABASE NEW friends\_db;
* START friends\_db DATABASE;
* CREATE DATABASE friends\_db;
* INIT friends\_db DATABASE;

18. When modifying data in a table, failure to use the WHERE statement will \_\_\_.

* prevent location services
* modify all the records
* stop the code from executing
* drop the database connection

19. Programmers use CRUD methodology which stands for \_\_\_.

* COMMON RECURSIVE ULTERIOR DESIGN
* CREATE READ UNDERSTAND DEFEAT
* CREATE READ UPDATE DELETE
* COMMON RECURSIVE UNIVERSAL DATA

20.  Data may be added to a specified table by using the \_\_\_ statement.

* INSERT INTO
* ADD INTO
* INSERT IN
* ADD DATA

21. The \_\_\_ operator can be used to select values within a range.

* RANGE
* BETWEEN
* VALUES
* MINMAX

22. The MySQL NPM package's createConnection method requires:

* user name and password
* database name
* host name
* all of the above

23.  Data returned from the database to a Node query is in the form of \_\_\_.

* an array of objects
* a single object
* a single array
* an array of arrays

24. The specific data being added to a table is preceded by the \_\_\_ keyword.

* VARIABLES
* VARCHAR
* DATA
* VALUES

25. The \_\_\_ statement is used to group together elements with shared values.

* GROUP BY
* SHARE FROM
* COMBINE INTO
* COMMON VALUE

26. In the code below, what would be placed in the <String> section?

connection.connect(function(err) {

if (err) throw err;

console.log("connected as id " + connection.threadId);

connection.query(<String>, function(err, res) {

if (err) throw err;

console.log(res);

});

});

* Name of the host, user and password
* SQL code for Node to send to the database
* Name of the database
* All of the above

27. In the prior question, connection.query is placed inside the connection.connect method's callback function so that the \_\_\_.

* integrity of the data is not compromised
* query will run only after the connection is made
* code is more readible and maintainable
* data can validated before sending it back

28.  Databases store data in one or more \_\_\_ comprised of multiple \_\_\_.

* containers, rows
* tables, columns
* arrays, variables
* variables, strings

29. The \_\_\_ data type can be used to specify that a column may only hold strings of variable lengths.

* STRING
* BLOB
* VARCHAR
* CHAR

30. The SQL \_\_\_ statement is used to select a database and perform SQL operations into that database.

* CREATE
* USE
* JOIN
* SELECT

0%

Bottom of Form