CHALLENGEs are OPTIONAL

00. Create directory panda/your\_name/hw/hw-2024-06-20

00. Create directory panda/your\_name/projects/<your-nodejs-mysql>

00. Create directory panda/your\_name/projects/09-calculator etc.

0. Projects

00-portfolio - get some ideas, what do you want

00-tic-tac-toe - 30.04.2024

01-is-it-a-prime-number - 30.04.2024

02-css-explanator - 30.04.2024

03-questionnaire - 8.05.2024

04-binary-guessing - 16.05.2024 - small groups

05-string-games - "fix the clock","random quote",

"palindrome", "gematria", "anagrams" parts

and the last part - "words-in-word" - finish the project - 26.06.2024

06-nim - 23.05.2024 - small groups

07-hangman - 5.06.2024 - small groups

with both "mouse click" and "keyboard" events,

with minimum of the listeners - using

event propagation

08-NodeJS-SQL-Your-Theme - continue with your theme

in accordance to the cookbook02.

See exercises 1. and 2. below.

19.06.2024

09-calculator - 16.06.2024 - build it with css GRID,

use both "mouse click" and "keyboard" events,

with minimum of the listeners - using

event propagation.

It should perform the actions: +,-,\*,/,%,power,

root, factorial, translating number from binary

to decimal, from decimal to binary, from hexadecimal

to decimal, from decimal to hexadecimal.

Optionally: translating numbers from decimal to

roman and from roman to decimal.

Optionally - more functions.

This project will be evaluated for the themes of:

"algorithms and advanced DOM".

Optional Projects:

- Canvas "Breakout Game"

- "Endless Dangers" - see the description below

- "Dreidel"

- "Memory Game" - see the beginning files and

the example of how to flip card inside

the "projects/optional/memory-game" dir

- "Math Practice" - Optional project of math-practice

with drops containing math exercises and

going down.

When the player types right answer,

the drop should stop or disappear or

make some other response.

- "One page site with smooth scrolling" - like YummyIsrael

1. Regular Expressions and Array Processing.

let words = ['cat', 'attempt', 'tattle', 'cattle', 'rate']

a) Create new array wordsTT that will contain all the elements

from "words" which contain 'tt'

b) Check if all the elements of "words" contain "at"

(pay attention: new array function,

the HINT is below all the exercises)

c) Check if any element of "words" contains "stat"

(pay attention: new array function,

the HINT is below all the exercises)

d) Take the array ['Cat', 'cot', 'CATER', 'SCat', 'ScUtTLe']

as is (without declaring it, just take it

as is) and filter it for the elements that contain "cat"

in any case.

(the HINT is below all the exercises))

e) Evaluate the next expressions without running them:

i) console.log('1,2,3,4'.replace(/,/, '-'))

ii) console.log('1,2,3,4'.replace(/,/g, '-'))

iii) console.log('cArT PART tart mArt'.replace(/art/ig, '2'))

iv) let word = 'cater'

console.log(word.replace(/cat/, 'hack'))

console.log(word)

v) word = word.replace(/cat/, 'hack')

console.log(word)

2. React. Use Counters.

a) HW MISSION from App.jsx

b) HW MISSION2 from App.jsx

3. Recursion.

a) Write a recursive function digitSum()

that gets a number and finds the sum of it's digits

but if the sum is not one digit, it again sums

the digits, etc. until it eventually will be

one digit.

b) Write function that checks recursively

if the given string is a palindrome

c) Use document.createElement, etc. and recursion

to create recursion-pic01.jpg

4. Callbacks and asynchronous.

a)Schedule Car Service (to run in a browser)

--------------------------

i) Create an asynchronous function

scheduleCarService(carModel, serviceType, callback)

that gets car model, service type,

and callback as parameters.

It should show:

"Thank you! Your <carModel>'s <serviceType> service

has been scheduled."

After that it should run setTimeout() for 1500 msec and show

after that:

"We're starting to perform the service ..."

After that it should run another setTimeout for 1000msec and

inside it's callback create the message:

let msg=`Your ${serviceType} service for ${carModel} is now complete`

and also inside run the callback that the function orderBook() got,

with an argument of the message (callback(msg)).

ii) Create 3 functions that can be used as callbacks for orderBook()

function.

1st. showInConsole(message) - shows the message in the console

2nd. showAlert(message) - shows the message with alert()

3rd. showInHTML(message) - shows the message like

document.body.innerHTML += `<p>${message}</p>`;

iii) Run scheduleCarService() thrice - first time pass it

showInConsole as a third argument,

like scheduleCarService("Toyota Camry", "oil change", showInConsole);

for the second time pass it showAlert as a third argument

and for the third time pass it showInHTML as a third argument

b) Unite file data. Run in NodeJS:

----------------------------------

i) Create asynchronous function

readTwoFiles(file1Path,file2Path,callback).

It should read 1st file into data1 and in

the case of success read 2nd file into data2

and in the case of success run the callback and

to pass to it data1 and data2.

ii) Create function uniteTwoFiles(file1Path,file2Path).

It states: "Going to unite <file1> and <file2> ..."

After that it calls readTwoFiles() and passes

it the path of the 1st file and the path of the 2nd file

and callback that is an arrow function with gets

2 strings and writes them into 1 file and

tells if it was successful or not.

HINT for 1 b): use words.every()

HINT for 1 c): use words.some()

HINT for 1 d): ['Cat', 'cot', 'CATER', 'SCat', 'ScUtTLe'].filter()