CHALLENGEs are OPTIONAL

00. Create directory panda/your\_name/hw/hw-2024-06-27

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. Refactor your "Is it a prime number?" project into React app.

2. Only for those who want to train more React:

Use Cookbook04 - of the TODO app and create your own

list (of jokes, movies, etc. that you're able to

cross out, to delete and to add) - don't make it more

complicated than TODO was, as we're going to make

more complicate things anyway.

3. Recursion. Continue from the lesson:

Create showFibonacci function that creates the string

comprised of the Fibonacci sequence elements till the given

element number. For example, showFibonacci(2) should show

1,1

and showFibonacci(7) should show

1,1,2,3,5,8,13

4. Recursion. Let's invent sequence almoust like Fibonacci,

but it's elements starting from the 4th will be sum of the

previous 3 elements.

1,1,1,3,5,9,17,31,57,105,193 etc.

a) Create function sequence(n) that shows the n-th element of

our sequence.

b) Create function showSeq(n) that shows all the elements of

the sequence till n-th element (including)

5. React Challenge: Perfect Gallery component

- Should be comprised of small components of 1 figure/article/div/...

- Should get data from an array of objects

- Should be responsive (use flex or grid ...)

- Optional: amount of items on the page

- What happens when we click on one item:

-- it is enlarged and what else?

-- does it have arrows to move between the enlarged items?

-- does it have button bar below it?

-- is it modal?

-- how it disappears? with "Esc"? with "x"?

- Could be sorted?

- Optional: automatic movement

6. React Challenge: Eratosthenos Sieve

For example: https://en.wikipedia.org/wiki/Sieve\_of\_Eratosthenes

-- Each number should be a component

-- Optionally: input of the max number

-- Show the cross-overs of the numbers in real time:

a) when the Sieve component mounts (useEffect)

b) with buttons "Start" and "Stop"

(not necessarily useEffect())

7. Optionally: convert more projects (like Nim, Binary Guessing,

String Games - to React component) - you could use them

in our future React project

8. RegEx - reminder. The solutions are below all the exercises.

a) const ar1 = ['Dany likes green apples more than the red ones',

'Red Hat - is it a fairy tale or LINUX name?',

'Red apple - I am pretty sure, that it was the reason, why Newton has discovered the gravity',

'"Bireishit" company stores yellow apples in special humid conditions']

Go over the array ar1.

If string contains "red apple" or "green apple" in any case - return true, otherwise - false.

b) const ar2 = ['Don\'t say it','You don\'t say it!','Don\'t do this, you\'ll regret it',

'Don\'t go there','Go there and see yourself','We don\'t know',

'They don\'t know, don\'t they?','Don\'t stay here and don\'t go there','Why don\'t you know?']

Go over the array ar2.

If string contains "don\'t say" or "don\'t go" or "don\'t know" in any case -

return true, otherwise - false.

c) const ar3 = ['His ear is aching','tear','bear','smear','wear','swear','Teddy-Bear',

'Bears run fast', 'Granma, why do you have these big ears?']

Go over the array ar3.

If string contains "ear" with any english letter before - return true, otherwise - false.

d) Go over the array ar3.

If string contains "ear" without "t" or "b" right before it, return true, otherwise - false.

e) Create a function noOther(ar) that gets a string array,

goes over it and returns true if no element in the array

contains no other characters except of english letters,

',','-','?',double quotes and spaces.

When called with ar1 or ar3 - this function should return

true, when called with ['U 2','4 me','Like',':)'] or ar2

it should return false.

9. Regular Expressions (new):

The RegExp object can be saved in a variable.

This helps to improve code clarity, enables reuse, etc.

> const pet1 = /dog/

> pet1.test('They bought a dog')

< true

> pet1.test('A cat crossed their path')

< false

RegExp objects can also be constructed using the RegExp() constructor. The first argument is either a string or a RegExp object. The second argument is used to specify one or more flags.

(We've already discussed that "constructor" is a function that

creates object of some class. Till now we've mostly used

the Date() constructor)

> const pet2 = new RegExp('tarantula')

> pet2

< /tarantula/

// if flags are needed, specify them as the second argument

> const pet3 = new RegExp('tarantula', 'i')

> pet3

< /tarantula/i

The main advantage of the constructor

over // literal is the ability to dynamically construct

the regexp using ${} to insert content of other variables

or the result of an expression.

> let greeting = 'hi'

> const pattern1 = new RegExp(`${greeting} there`)

> pattern1

< /hi there/

> const pattern2 = new RegExp(`${greeting.toUpperCase()} there`)

< /HI there/

> const pattern3 = new RegExp(`${greeting} there`,'ig')

> pattern3

< /hi there/ig

a) const ar4 = ['My cat likes to play with my TARANTULA',

'Dogs and Tarantulas are the best friends of a man',

'When dog barks all the cats smile',

'The best spider in the world is tarantula']

Go over ar4 and replace pet1 pattern by 'dolfin',

pet2 pattern by 'whale', pet3 pattern by 'bear'.

b) const ar5 = ['01110101','A010101','10021001','10001000']

Create pattern4 using only literal assignment like

patternx = /[a-c]$/ - to check that a string does not contain

other characters except of '0' and '1'.

c) const ar6 = ['Good Morning to you all!','What a sunny day!',

'Beautiful sunny evening','Is this still a day?',

'On this sunny morning we'll start our tour and will end it blessing you with "Good Evening!"']

Use RegExp constructor and create regular expression

in such a way that you will replace with it only the first

occurences of "good day","good morning","sunny day",

"good evening", "sunny evening","sunny morning"

- in any case - with "cloudy night" .

10. Create a class of University.

It's constructor should get a name of a city,

a name of university, max number of faculties,

and an object of faculties, like:

{

Mathematics: {dean: 'Moshe Baruch',students:189},

Physics: {dean: 'Adi Naser',students:132},

Biology:{dean:'Michael Glikman',students: 176},

ComputerSciences:{dean:'Dani Raz',students:512},

Architecture: {dean:'Gadi Kapluto',students:273},

Medicine:{dean: 'Ami Aharonheim',students: 211},

DataAndDecisionSciences: {dean: 'Ran Smorodinskii', students: 603}

}

This class should have the next methods:

a) aboutUniversity() - it should show which faculties

the univercity has and how many students in total are there.

It also creates new property "totalStudents" or updates it.

b) acceptStudent() - it should get a name of a student

and a name of a faculty and add 1 to the number of the students

of the faculty and to show

"Congratulations, <student name>, our dean <dean name>

blesses you with you being accepted to our faculty of

<faculty name> of the glorious <university name> in our

beautiful city of <city name>"

c) graduateStudent() the same as b), just minus 1

(check, that there are still some students)

d) buildNewFaculty() - it should get faculty name

and dean name,check if there still place for one

more faculty and set 0 students

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

i) Create Technion object (with the above faculty object)

and some other university object.

ii) Try your methods on the both objects.

11. Number 3. from the previous Homework - updated:

Create choose-a-user-updated React app.

See HW\_useContext\_example2.png

a) Get App-choose-a-user-updated.jsx

b) You may distribute the components into different files or leave them in the same file

c) Convert userName of App to state

d) Create Context and add userName and users to the Context

e) At UserInfo get userName and users from the Context and show the info about the user

f) Create in App function changeUser(newName) that changes the userName and add it to the context

g) Make SelectUser component not hard-coded but get from the context

"users" and build it

h) Get the function changeUser from the Context at SelectUser and use it onChange

to change the current user and it's info

i) In the About component get userName and "users" from the context,

use current user favorite color for the background instead of 'yellow',

use current user favorite book instead of "Diary of a Wimpy Kid",

create the list of the current user's hobbies in the <ul> instead of all the "knitting"

j) In the UserInfo component get userName and "users" from the context,

use there current user's nick and name (in the brackets),

pass the right props to the Emoji component

Solutions for 8:

a)const ar1Result = ar1.map(str => /(red|green) apple/i.test(str))

ar1Result: [true, false, true, false]

b)const ar2Result = ar2.map(str =>/don\\'t (say|go|know)/i.test(str))

ar2Result: [true,true,false,true,false,true,true,true,false]

c) const ar3Result = ar3.map(str => /[a-z]ear/i.test(str))

ar3Result: [false,true,true,true,true,true,true,true,false]

d) const ar3Result2 = ar3.map(str => /[^bt]ear/i.test(str))

ar3Result: [false,true,true,true,true,true,true,true,false]

e) function noOther(ar) {

return ar.every(el => /[^a-z, "?\-]/i.test(el))

}

noOther(ar1) // true

noOther(ar2) // false

noOther(ar3) // true

noOther(['U 2','4 me','Like',':)']) // false