CHALLENGEs are OPTIONAL

00. Create directory panda/your\_name/hw/hw-2024-07-10

00. Create directory panda/your\_name/projects/10-hanoy-towers

00. Create directory panda/your\_name/projects/11-shopping-carts

00. Create directory panda/your\_name/projects/12-game-portal

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".

10-hanoy-towers - finish tbd + improve +

see recursion-algorithm.txt and calculate the max number of steps for the player

+ challenge - create recursive movement of the disks - 17.07.2024

11-shopping-carts - the requirements are in

projects/11-shopping-carts/about-shopping-carts.txt - 17.07.2024

12-react-game-portal - the requirements are in

projects/12-react-game-portal/about.txt - 26.07.2024

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. 10-hanoy-towers - finish tbd + improve +

see recursion-algorithm.txt and calculate the max number of steps for the player

+ challenge - create recursive movement of the disks - 17.07.2024

2. 11-shopping-carts - the requirements are in

projects/11-shopping-carts/about-shopping-carts.txt - 17.07.2024

3. Go up over the "react-levels"

4. extends. Take the script extends2.js.

Create one user, show that he can login adn logout.

Create class Player that extends class User.

Class Player should have number properties: level, score, health, strength.

Class Player should have string property: typeOfPlayer - it can be

'Archer','Warrior','Mage','Healer'.

It should have methods of addPoints(n) - if the points get to some

number that is dividible by 10, level up,

showUser() - shows it's type, score, level, health.

Create one player, show that he can login, logout, get more score.

Create class Healer which extends class Player,

it should has method of heal(player) -

adds 5 points to the health of the given player.

Create one healer, use with him showUser() method.

Use also heal() method - on the previous player.

5. clean-code.io - preparation for JS exam (that should be

on 16.07.2024) - the themes are:

nested setTimeout()

clearTimeout()

setInterval()

clearInterval()

nested ternary operator

rest operator

spread operator

arrays - destructuring

objects

objects - destructuring

async including setTimeout, setInterval and async callbacks

Classes (only what we've learnt till now - no setters/getters)

regex (only what we've learnt till then)

recursion (don't worry, not serious)

6. Regular Expressions. See the solutions below.

New: ranges could be shown not only as

a-t or 3-9, for example, but also by special characters.

\d - Matches any digit character. Equivalent to [0-9].

\w - Matches any word character, where a word character

includes letters (A–Z, a–z), numbers (0–9), and underscore (\_).

If the regex is Unicode-aware and the i flag is set,

it also matches other Unicode characters

that get canonicalized to one of the characters above through case folding.

\s - Matches any whitespace (space, tab, new line) or line terminator character.

The uppercase forms \D, \W, and \S create complement character classes

for \d, \w, and \s, respectively. They match any character

that IS NOT IN THE SET of characters matched by the lowercase form.

a) Invent a regular expression to match any non-word character.

Test strings: ["/","7","%%","%","\_","T"]

Should return: true,false,false,true,false,false

b) Create a regular expression that matches a string

starting with a non-letter character and ending with some word character.

Test Strings: ["-Hello","3 brothers","<span>","<ul> - is an HTML tag"]

Should return: true,false,false,true

c) Split the string by any whitespace characters:

let str1 = `Don't go there-it's\tscary.

Really?`

Should return: ["Don't","go","there-it's","scary.","Really?"]

d) Now split the new string not only by whitespace characters, but also

by "," "-" "." and "?"

let str2 = `How would you catch a spider?No,I wouldn't-why should\tI

So,don't`

Should return: ['How', 'would', 'you', 'catch', 'a', 'spider',

'No', 'I', "wouldn't", 'why', 'should', 'I', 'So', "don't"]

Solution of 6:

a) Regular Expression: /^\W$/

b) Regular Expression: /^\W\w\*/

c) str1.split(/\s/)

d) str1.split(/[,?.-\s]/)