CHALLENGEs are OPTIONAL

00. Create directory panda/your\_name/hw/hw-2024-06-16

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" and "gematria" parts - 30.05.2024

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. In parallel with our project Yummy

continue your own project

08-NodeJS-SQL-Your-Theme -

show the page with the form for adding an entity.

Use Cookbook002 until step 45 including.

This project in the future is getting evaluated for the theme of

"Basic NodeJS Express MVC CRUD API"

2. Continue the project Yummy and finish the parts

of "editFood" and "deleteFood" -

go over the new steps in the Cookbook02,

starting from the step 46.

Also finish these parts in your own project.

On 19.06.2024 we are going to learn to upload it.

OPTIONAL additions:

a) while performing editFood - to replace the image

b) while performing deleteFoor - to show some

"Are you sure? ..." screen or alert and to enable

the user to cancel the deletion or to proceed with it.

c) add options to add/edit/delete cuisine

3. SQL - JOINS:

a) Run the script SQL-examples4-whole-classicmodels.sql -

it will recreate the old and create the new tables

in the database `classicmodels`

b) Use table "orders" and table "customers"

i) Show orderDate, shippedDate, status and customerName

ii) Get only those where status is "shipped" and order them

by shippedDate from the recent backwards.

iii) Instead of ii) get only those where status is not shipped

and order them by status and by customerName

iv) Add to the previous condition to get only the orders that their

orderDate is in the range from 2004-05-07 till 2005-01-12

including

c) Use table "orders" and table "orderDetails"

i) show orderDate, shippedDate, status, customerNumber, productCode, quantityOrdered, priceEach

and order by orderNumber

ii) add column that is a result of multiplication of quantityOrdered by priceEach,

rounded to zero digits after a dot, and give it a name "total"

iii) inside each order set the rows from the biggest to the smallest "total"

iv) and show only the rows for wich the above calculation result is greater or

equal to 3223 and smaller than 3232

v) instead of iv) show only the rows that in their productCode have after "S"

and before "underscore" number that is 700 or 25 or 50 or 32 or 72 -

use for this the functions SUBSTRING() and INSTR()

vi) and that their price is in the range from 60.9 to 65.15 including

vii) and that the number after "underscore" in their productCode is bigger or equal to 2000

(use INSTR and SUBSTRING)

d) Use table "customers" and table "employees"

i) show contactFirstName, contactLastName, and from "employees":

concatenate with spaces firstName, lastName and jobTitle

and call this column "full\_name"

ii) show only the rows where contactFirstName and sales representative

first name have the same first letter

4. Callbacks and asynchronous.

You can run these by NodeJS or in a browser.

a) Random Joke Generator

Create function badJoke() that gets a joke as a parameter,

says "Here is your random bad joke:" and shows the joke

in the console.

Create function funnyJoke() that gets a joke as a parameter,

says "Here is your random hilarious joke:" and shows the joke

in the console.

Create function shortJoke() that gets a joke as a parameter,

says "Here is your pretty short random joke:" and shows the joke

in the console.

Use the following function:

function generateJoke(callback) {

const jokes = ['I went shopping for a pair of camouflage pants. But I couldn't find any.',

'My dad was hit on the head with a can of soda. Luckily, it was a soft drink.',

'I used to steal soap, but I'm clean now.',

'Why do tigers have stripes? They don't want to be spotted.']

console.log(Date.now());

console.log('Generating a random joke...');

setTimeout(() => {

let joke = jokes[Math.floor(Math.random()\*jokes.length)];

console.log(Date.now());

callback(joke);

}, 2500);

}

Now run generateJoke 3 times: pass to it as a callback

badJoke, funnyJoke and shortJoke.

b) Weather Forecast

function showTime() {

const currDateTime = new Date();

console.log(currDateTime.getHours() + ":"

+ currDateTime.getMinutes() + ":"

+ currDateTime.getSeconds() + ":"

+ currDateTime.getMilliseconds());

}

Write an asynchronous function `weatherForecaster()`

that takes a callback function, a city and a delay as parameters.

Inside the function, display the current time using

showTime(), print a message about fetching the weather forecast

for the given city like

`In 1500 milliseconds going to fetch

the weather forecast for Tzfat ...`,

and use setTimeout() to simulate an asynchronous delay.

Pass the callback and a specified delay time as arguments to

`setTimeout()`.

Use the following functions:

```

function fetchEilatWeather() {

showTime();

console.log(`Fetching weather forecast for Eilat`);

setTimeout(() => {

showTime();

console.log(`Weather forecast retrieved for Eilat`);

}, 3000);

}

```

```

function fetchTzfatWeather() {

showTime();

console.log(`Fetching weather forecast for Tzfat`);

setTimeout(() => {

showTime();

console.log(`Weather forecast retrieved for Tzfat`);

}, 3000);

}

```

Call `weatherForecaster()` and pass it `fetchEilatWeather()`

along with 'Eilat' and a delay time.

Call `weatherForecaster()` again and pass it `fetchTzfatWeather()`

along with 'Tzfat' and a delay time.

c) Weather Forecast For Any city

Write an asynchronous function `weatherForecasterForAnyCity()`

that takes a callback function, a city and a delay as parameters.

Inside the function, display the current time using

showTime(), print a message about fetching the weather forecast

for the given city like

`In 1500 milliseconds going to fetch

the weather forecast for Tzfat ...`,

and use setTimeout() to simulate an asynchronous delay.

Pass an arrow function that runs the callback

with the specified city and a specified delay time

as arguments to `setTimeout()`.

```

function fetchWeather(city) {

showTime();

console.log(`Fetching weather forecast for ${city}`);

setTimeout(() => {

showTime();

console.log(`Weather forecast retrieved for ${city}`);

}, 3000);

}

```

```

function fetch2DayWeather(city) {

showTime();

console.log(`Fetching 1st day weather forecast for ${city}`);

setTimeout(() => {

showTime();

console.log(`1st day weather forecast retrieved for ${city}`);

console.log(`Fetching 2nd day weather forecast for ${city}`);

setTimeout(() => {

showTime();

console.log(`2nd day weather forecast retrieved for ${city}`);

}, 2000);

}, 2000);

}

```

Call `weatherForecaster()` and pass it `fetchWeather`

along with a city and a delay time.

Call `weatherForecaster()` again and pass it `fetch2DayWeather`

along with a city and a delay time.