JSF – Part 1
Exercises

The contained in this document exercises are related to the JavaScript Foundations – Part1 training course. The exercises are an integral part of the JSF lectures, designed to train IT Technical Consultants.

The following naming convention is followed: the first word (the prefix) points to the material presented in the relevant lecture, followed by a subsequent tasks' numbers.

Table of Contents

Document History

Document version: **v.1.0**
Document date: **25.04.2024**
Initial version

Document version: **v.1.1**
Document date: **26.04.2024**
Added: BASICS03-001 – BASICS03-020

Document version: **v.1.2**
Document date: **29.04.2024, 23:22h**
Added: BASICS02-014, BASICS02-015, BASICS03-021, BASICS03-022
Added Document History
Added Tasks/Exercises counters.

Document version: **v.1.3**
Document date: **08.05.2024, 21:51h**
Added: DTYPES01-001 – DTYPES01-023

Document version: **v.1.4**
Document date: **20.05.2024, 22:30h**
Updated: DTYPES01-007 (changed task)
Updated: DTYPES01-009 (changed task)
Updated: DTYPES01-015 (changed task)
Added: DTYPES01-024 – DTYPES01-025

Document version: **v.1.5**
Document date: **20.05.2024, 22:30h**
Updated: BASICS02-005 (clarification)

Document version: **v.1.6**
Document date: **07.06.2024, 01:50h**
Added: DTYPES02-001 – DTYPES02-008

Document version: **v.1.7**
Document date: **11.06.2024, 00:33h**
Added: DTYPES02-009 – DTYPES02-012

Document version: **v.1.8**
Document date: **14.06.2024, 02:45h**
Added: LOOPS01-001 – LOOPS01-013

Document version: **v.1.9**
Document date: **27.06.2024, 21:00h**
Added: ARRAY01-001 – ARRAY01-023

Document version: **v.1.10**
Document date: **04.07.2024, 01:00h**
Added: FUNCTIONS01-001 – FUNCTIONS001-011
Added: FUNCTIONS02-001 – FUNCTIONS002-003
Added: FUNCTIONS03-001 – FUNCTIONS003-003

Document version: **v.1.11**
Document date: **12.07.2024, 02:30h**
Added: FILES01-001 – FILES01-002

Tasks/Exercises Count

BASICS01	11 tasks
BASICS02	15 tasks
BASICS03	22 tasks
DTYPES01	25 tasks
DTYPES02	12 tasks
LOOPS01	13 tasks
ARRAY01	23 tasks
FUNCTIONS01	11 tasks
FUNCTIONS02	3 tasks
FUNCTIONS03	3 tasks

Basics 01 – Names, Variables

BASICS01-001: Write ten correct identifiers, following the camelCase naming convention.

BASICS01-002: Imagine, you are solving a math problem. Declare ten variables, which you might need in your program.

BASICS01-003: Imagine, you are working for a cloud provider and are responsible for the servers. You must write a program to list and describe the servers. Declare ten variables, which you might need in your program.

BASICS01-004: Print on the console five alphabet characters, five numbers, five punctuation characters.

BASICS01-005: Declare five variables, assign some numbers, and print them on the console.

BASICS01-006: Declare ten variables, assign the numbers from 1 to 10 and print the even numbers on the console.

BASICS01-007: Declare ten variables, assign the numbers from 1 to 10 and print the first three odd numbers on the console.

BASICS01-008: Declare ten variables, assign the numbers from 100 to 109 and print the last two odd numbers on the console.

BASICS01-009: Declare five variables, assign the first five prime numbers, and print them on the console.

BASICS01-010: Declare ten variables. On the first five - assign the first five prime numbers. On the second five numbers, do the same, but multiply each value by 3. Print all of them on the console.

BASICS01-011: Declare ten variables. Assign them the first ten prime numbers. Print the numbers in reverse order.

Basics 02 – Character Set

BASICS02-001: Declare five variables. Assign them with the ASCII codes of the first five English capital alphabet characters. Print them on the console.

BASICS02-002: Declare five variables. Assign them with the ASCII codes of the last five English lowercase alphabet characters. Print them on the console.

BASICS02-003: Declare five variables. Assign them with the ASCII codes of randomly chosen punctuation characters. Print them on the console.

BASICS02-004: Declare five variables. Assign them with the UNICODE codes of randomly chosen emoji characters. Print them on the console – on different lines.

BASICS02-005: Declare five variables. Assign them with the UNICODE codes of randomly chosen emoji characters. Print them on the console on one line with four spaces between them.

BASICS02-006: Declare five variables. Assign them with the randomly chosen emoji characters. Print the UNICODE codes on the console – on different lines.

BASICS02-007: Declare five variables. Assign them with the randomly chosen emoji characters. Print the UNICODE codes on the console – on one line, separated with commas and space after each comma character.

BASICS02-008: Declare five variables. Assign them with the randomly chosen emoji characters. Print the UNICODE codes in hex format on the console – on different lines.

BASICS02-009: Declare five variables. Assign them with the randomly chosen emoji characters. Print the UNICODE codes in decimal format on the console – on different lines.

BASICS02-010: Declare five variables. Assign them with the randomly chosen emoji characters. For each of the variables - print the UNICODE code in binary, octal, decimal, and hex format on one line, separated with comas and space after it.

BASICS02-011: Declare two variables. Assign them with two English capital alphabet characters. Compare them with the “lower than” operator (<) and print on the console the result.

BASICS02-012: Declare two variables. Assign them with two English alphabet characters – one in capital and the other one in lowercase. Compare them with the “lower than” operator (<) and print on the console the result. Can you describe the result?

BASICS02-013: Declare two variables. Assign them with one English alphabet character and one number character. Compare them with the “greater than” operator (>) and print on the console the result. Can you describe the result?

BASICS02-014: Declare a variable. Assign one letter from the English alphabet. Print on the console the reverse – if the letter is capital, print it in lowercase; if the letter is in uppercase, print it in lowercase. Hint: Use the encoding table/codes/location in the table.

BASICS02-015: Declare a variable. Assign it with a letter from the English alphabet. Define a constant, named Cipher and assign it a value in the range of [3;13]. Print on the console the Cipher-th letter after the assigned letter to the variable. Imagine that the English

alphabet is linked in a circle (cycle) – after Z letter follows A letter, then B etc. Example: if you have Cipher=4, and the given letter is 'Y', then print on the console 'C' (the fourth after 'Y').

Basics 03 – Operations, Operators, Precedence

BASICS03-001: Declare two variables and assign them two integer numbers. Print on the console the result of their division.

BASICS03-002: Declare two variables and assign them two integer numbers. Print on the console the division remainder (modulus - остатък от целочислено деление).

BASICS03-003: Declare four variables. On two of them assign integer numbers. The third set with the division remainder. The fourth one set with the quotient (частното -> цялата част от делението). Print on the console the four variables with appropriate description.

BASICS03-004: Define a constant. Check and print on the console if the constant is positive, negative or zero. Hint: Use ternary operators. How many operators do you need?

BASICS03-005: Declare three variables and assign them with three randomly selected integer numbers. Print on the console those two of them, which have the biggest sum. Hint: Use the ternary operators.

BASICS03-006: Declare one variable, assign integer number. Check if the variable contains an even number. Print on the console appropriate message.

BASICS03-007: Declare a constant and assign one digit. Print on one line the constant, the power of two (N^2), the power of three (N^3) on the console.

BASICS03-008: Declare a variable. Assign one digit from the range of [1;9]. Print on the console the multiplication table with that variable.

BASICS03-009: Calculate and print on the console the perimeter of a triangle.

BASICS03-010: Calculate and print on the console the surface area (лицето) of a triangle. ($S=(a.h_a)/2$)

BASICS03-011: Calculate and print on the console the perimeter of a rectangle.

BASICS03-012: Calculate and print on the console the surface area of a rectangle.

BASICS03-013: Calculate and print on the console the perimeter (the length) of a circle. ($C=2.\pi.r$)

BASICS03-014: Calculate and print on the console the surface area of a circle.

BASICS03-015: Declare a variable. Assign one digit from the range of [1;9]. Print on the console the multiplication table with that variable, but in reverse order – first print the multiplication with 10, then with 9, etc. Also, print on each line the calculated result from the multiplication raised to the second power (на степен 2).

BASICS03-016: A bus leaves from point A to point B with speed of 80 km/h. At the same time, a car leaves from point B to point A with speed of **x** km/h. The distance between point A and point B is **S** kilometers. After how many minutes, the bus, and the car will meet? Print the result on the console. (S = V.t)

BASICS03-017: Write a JavaScript program to convert degrees in radians. Print on the console an appropriate message.

BASICS03-018: Write a JavaScript program to convert km/h into km/min. Print on the console an appropriate message.

BASICS03-019: Write a JavaScript program to convert km/h into m/s. Print on the console an appropriate message.

BASICS03-020: Declare a variable. Assign an integer number. Print on the console the variable, the binary, octal and hexadecimal representation.

BASICS03-021: Are there any not correctly defined expressions in the following excerpt? If any – which one(s)? Why?

```
let a = 1;
let b = a;

let r1 = a+--b;
let r2 = a+++b;
let r3 = a---b;
let r4 = a-++b;
let r5 = a*++b;
let r6 = a*--b;
let r7 = a**++b*b;
let r8 = a**++b                    //b;
```

BASICS03-022: You have the following excerpt of a JavaScript code:

```
let a = 1;
let b = 3;
let result = a**++b/b++**a;
```

Try to calculate the value of the result variable, without executing the code.

DTypes 01 – Strings

DTYPES01-001: Define a string variable with your names (first and last) in two ways (hint: use different quotes).

DTYPES01-002: Declare two variables. Initialize one of them with your first name and the other one – with your last name. Define a variable called **fullName** with a template string. Use variable substitution for the two simple names and use a space character as a separator. Print fullName on the console.

DTYPES01-003: Declare two variables. Initialize one of them with your first name and the other one – with your family name. Define a variable called **fullRevName** with a template string. Use variable substitution for the two simple names, having the family name as a first name. Use comma character as a separator. Print fullRevName on the console.

DTYPES01-004: Define two multi-line string variables in three ways. (hint: use single quotes, double quotes, string literals).

DTYPES01-005: Define a string variable with following format:

My name's <name>. I'm <age> years old.

Previously define the name and the age values. They will not be changed during the further code.

DTYPES01-006: Define four string variables with your three names (first, middle and family). The fourth variable initialize with all names, separated with a tab character. Print on the console the length of each variable.

DTYPES01-007: Define four string variables with your three names (first, middle and family). The fourth variable initialize with all names, separated with a dash "-" character. Make sure that only the first letter of each name is capital, all other letters are lowercase. Print on the console the length of each string.

DTYPES01-008: Define four string variables with your three names (first, middle and family). The fourth variable initialize with the concatenation of the first characters from each name. Print the fourth variable on the console.

DTYPES01-009: Define four string variables with your three names (first, middle and family). The fourth variable initialize with the concatenation of the first characters from each name (capitalize them), separated with a dot "." character. Print the fourth variable on the console.

DTYPES01-010: Define four string variables with your three names (first, middle and family). The fourth variable initialize with the concatenation of the first characters (capitalize

them) from each name plus the concatenation in reverse order (again capital letters). Print the fourth variable on the console.

DTYPES01-011: Define three string variables with your three names (first, middle and family). Declare a fourth variable. Initialize it with the sum of the character codes from the first characters from each name. Print all variables with appropriate text on the console.

DTYPES01-012: Define three string variables with your three names (first, middle and family). Declare a fourth variable. Initialize it with the sum of the character codes from the last characters from each name. Print all variables with appropriate text on the console.

DTYPES01-013: Define three string variables with your three names (first, middle and family). Declare a fourth variable. Initialize it with the sum of the character codes from the first characters from each name, minus the sum of the character codes from the last characters from each name. Print the fourth variable on the console.

DTYPES01-014: Define a string variable with the following text (called pangram):

The quick brown fox jumps over the lazy dog

Use positive indexing to print on the console the character "b".
Use negative indexing to print on the console the character "s".

DTYPES01-015: Define three string variables with your three names (first, middle and family). Declare a fourth variable. Initialize it with the sum of the character codes from the second characters from each name, minus the sum of the character codes from the characters before last from each name. Print the fourth variable on the console.

DTYPES01-016: Print the numbers from the following list: 1, 10, 38, 4, 824, 120, 999, 64 in one column, right justified. Pad all the digits with zeros so that the column is right-justified, and the width is exactly the width of the longest number written in the list. The length of a number is measured by the number of digits involved in its writing.

DTYPES01-017: Print the numbers from the following list: 1, 10, 38, 4, 824, 120, 999, 64 in one column, right justified. Pad all the digits with zeros so that the column is right-justified, and the width is exactly eight characters. Also, replace all eights with the digit nine, and all nines with the digit one.

DTYPES01-018: The following permanent text is given:

Primitive

Print on the console the following:
On the first line – print the first character.
On the second line – print first two characters.
On the third line – print first three characters and so on until the whole word is printed.

DTYPES01-019: The following permanent text is given:

Primitive

Print on the console the following:
On the first line – print the first character.
On the second line – print first two characters.

On the third line – print first three characters and so on until the whole word is printed. Pad each line with space character(s), so the text is right justified.

DTYPES01-020: Print all even (четни) numbers from the string:

The number 28469 is not so big.

Hint 1: Check the remainder of integer division by two.

Hint 2: Use the ternary operator.

DTYPES01-021: Count the number of characters in the English word for each digit ([0;9]). Print on the console, in different lines, and in comma separated format for each digit the following - the digit, the English word and the number of letters.

Example:

...

6, six, 3

7, seven, 5

DTYPES01-022: Check if the word “jump” exists in the following text (called pangram):

The quick brown fox jumps over the lazy dog

If it does not exist -> print “does not exist”.

If it is fount -> print “found at position: <position>.

DTYPES01-023: Check if the word “jump” exists in the following text (called pangram), ignoring the case sensitivity:

The quick brown fox jumps over the lazy dog

If it does not exist -> print “does not exist”.

If it is fount -> print “found at position: <position>.

DTYPES01-024: Count the number of occurrences of each digit in the following text:

“193817588”

Print in ascending order all numbers from 0 to 9 and the number of times they occur in the text, separated with a comma.

DTYPES01-025: Count the number of occurrences of each digit in the following text:

“19002288034”

Print in ascending order all numbers encountered in the string and the number of times they occur in the text, separated with a colon character “:”.

DTypes 02 – Numbers

DTYPES02-001: Define two number variables without decimal, two with decimal, two number variables in exponential notation; two variables in exponential notation (but with 4 digits for the whole part and 7 digits for the fractional part (остатък)).

DTYPES02-002: Try to re-write the following fragment of code:

```
let x = 0.12 + 0.21;
console.log(x);                    //x = 0.32999999999999996
```

to calculate **x** properly.
HINT: work with integer numbers and then use the right number of decimals.

DTYPES02-003: Make a program to add one variable of type Number with a small value and one variable of type BigInt also with a small value. Print the type of the result and the result on the console.

DTYPES02-004: Write a program in JavaScript to check if a given number is even or odd.
// use a ternary operator and string template

DTYPES02-005: Using Number and BigInt types, define variables and perform the actions with each of the arithmetic operations (+ - * /). Save the results in different variables and print them on the console.

DTYPES02-006: Add two big numbers (each with at least 30 digits). Print the result on the console.

DTYPES02-007: Multiply two big numbers (each with more than 20 digits). Print the result on the console.

DTYPES02-008: Define 5 bigint variables and print them in hexadecimal number system.

DTYPES02-009: Use a global number method and convert the Boolean value, representing a not-true value to a number and print on the console the calculation of the generated number raised to the power of itself. Print the result on the console.

DTYPES02-010: What is the result of the following arithmetic operations:

1) NaN++
2) NaN - NaN
3) Infinity - Infinity
4) 5 - true
5) Boolean(true-true)*true

DTYPES02-011: What will be the result, if you convert an empty string variable to Boolean and one more time convert the result to number? Print the result on the console.

DTYPES02-012: Multiply a variable (with nine digits, initialized with the sequential numbers from 1 to 9) from BigInt type with a variable, initialized as a number with value 2. Print the result on the console.

LOOPS 01 – If-else, For, While

LOOPS01-001: Write a program that determines whether a given number is positive or negative.

LOOPS01-002: Write a program that determines if a given number is even or odd.

LOOPS01-003: There are given two variables initialized with numbers. Write a program to determine which one is the greater.

LOOPS01-004: Write a program that calculates the Body Mass Index (BMI) and categorizes it. The formula for BMI is: weight / (height * height). Define three categories by your own discretion.

LOOPS01-005: You are doing an online test and at the end are receiving certain number of points. Write a program, that assigns a grade (string value) based on the received points from the test. Use the following grade scale:

Points from	Points to	Grade
0	19.99	Слаб 2 (F)
20	39.99	Слаб 2 (FX)
40	49.99	Среден 3 (E)
50	59.99	Среден 3 (D)
60	69.99	Добър 4 (C)
70	84.99	Мн. добър 5 (B)
85	100	Отличен 6 (A)

LOOPS01-006: Define a variable and initialize it with a number from the interval [0,23] (this will be interpreted as an hour). Write an appropriate congratulations message, based on the variable's value.

LOOPS01-007: Write a program to count the number of vowels (гласни) in a given string using for loop and if-else.

LOOPS01-008: Write a program to calculate the mathematical function factorial of a given number **n**.
0! = 1
1! = 1
2! = 1 . 2
3! = 1 . 2 . 3
n! = 1 . 2 . 3 . ... n

LOOPS01-009: Write a program to reverse a string.

LOOPS01-010: Write a program to check if a given number is prime (divisible only by 1 and itself).

LOOPS01-011: Write a program to check if a given string is a palindrome (reads the same string in both directions – from the beginning to the end and vice versa – the same string).

LOOPS01-012: Write a program to print the coordinates of a table cells in the format: (x, y). The table sizes will not be changed within the program.

LOOPS01-013: Like task LOOPS01-012 but use exactly 3 positions for each number in the coordinates and when printing them on the console, right-justify the numbers.

ARRAY 01 – one and two dimensional

ARRAY01-001: Write a program to find the **minimum element** from a one-dimensional array.

ARRAY01-002: Write a program to find the **maximum element** from a one-dimensional array.

ARRAY01-003: Write a program to find the difference between the **maximum element** and the **minimum element** from a one-dimensional array.

ARRAY01-004: Write a program to print only the positive elements from one-dimensional array.

ARRAY01-005: Write a program to print only the elements greater than or equal to a certain value, from one-dimensional array.

ARRAY01-006: Write a program to print the odd elements greater than a certain const value from one-dimensional array.

ARRAY01-007: Write a program to print the positive even elements, greater than a certain const value from one-dimensional array.

ARRAY01-008: Write a program to create and print one-dimensional array in reverse order.

ARRAY01-009: Write a program to find the average value from one-dimensional array, filled in with numbers. Create a new array, having the elements from the first one but each of them increased with the average value.

ARRAY01-010: Find the minimum elements from each column in two-dimensional array.

ARRAY01-011: Find the maximum elements from each row in two-dimensional array.

ARRAY01-012: Add two N x N arrays.

Hint: each element is the addition of the corresponding elements from both N x N arrays:
resultArr[0,0] = A[0][0] + B[0][0]

ARRAY01-013: Transpose a square matrix.

// [БГ]: Транспониране на матрица:
// редовете на **A** стават стълбове (колони) на $\mathbf{A}^T$;
// стълбовете (колоните) на **A** стават редове на $\mathbf{A}^T$.

ARRAY01-014: From a given table, find the row with the biggest sum of its elements.

ARRAY01-015: From a given table, find the column with the largest sum of its elements.

ARRAY01-016: From a given N x N table, find the sum of its two main diagonals .

ARRAY01-017: Find the first n-elements from Fibonacci sequence, having the first two elements 1 and 1, and every sequential element is the sum from the previous two. Put the elements in an array. Find their sum.

ARRAY01-018: Find the sum of all numbers in a table. Assume that the table contains only numbers.

ARRAY01-019: Count the even numbers in a two-dimensional array. Calculate the percentage of this amount from the total amount of numbers in the array.

ARRAY01-020: Count the amount of elements (numbes) in a two-dimensional array, who are bellow a certain threshold.

ARRAY01-021: Print on the console all elements (numbers) of a table (N x N), using exactly 5 positions for each element, right justified.

ARRAY01-022: Array A has N elements. The two-dimensional array B has N x N size. Check if the array A matches any of the rows in array B. Print the row number if matches, otherwise – print “no match with the rows”.

ARRAY01-023: Array A has N elements. The two-dimensional array B has N x N size. Check if the array A matches any of the columns in array B. Print the column number if matches, otherwise – print “no match with the columns”.

FUNCTIONS 01

FUNCTIONS01-001: Write a function that takes two numbers as arguments and returns the largest of them.

FUNCTIONS01-002: Write a function that takes a single number as an argument and returns true if the number is even, and false otherwise.

FUNCTIONS01-003: Write a function that takes a number as an argument and returns:

- **"Fizz"** if the number is divisible by 3
- **"Buzz"** if the number is divisible by 5
- **"FizzBuzz"** if the number is divisible by both 3 and 5
- The number itself if it is divisible by neither

FUNCTIONS01-004: Write a function that takes a string as an argument and returns the longest word in the string.

FUNCTIONS01-005: Write a function that takes a number as an argument and returns the sum of its digits.

FUNCTIONS01-006: Write a function that takes a string as an argument and returns the number of words in the string.

FUNCTIONS01-007: Write a function that takes three numbers as arguments and returns the biggest sum of two of the numbers.

FUNCTIONS01-008: Write a function that calculates the average value of a numbers in array

FUNCTIONS01-009: Write a function that takes a string and a character as arguments and returns the number of times the character occurs in the string.

FUNCTIONS01-010: Write a function that takes an array as an argument and returns a new array with all false values removed.

FUNCTIONS01-011: Write a function that takes a string as an argument and returns the string with the first letter of each word capitalized.

FUNCTIONS 02 – Nested Functions

FUNCTIONS02-001: Write a function that takes a nested array (an array containing arrays of numbers) and returns a new nested array with only the even numbers. Try to solve the problem with nested functions.

FUNCTIONS02-002: Write a function that takes a nested array (an array containing arrays of strings) and returns a single concatenated string of all the strings in the nested arrays. Try to solve the problem with nested functions.

FUNCTIONS02-003*: Write a function that takes a nested array (an array containing arrays of numbers) and returns the smallest and the largest number from all the nested arrays. Try to solve the problem with nested functions.

FUNCTIONS 03 – Recursive Functions

FUNCTIONS03-001: Write a recursive function that takes an array of numbers and returns the sum of all the elements.

FUNCTIONS03-002: Write a recursive function that takes a number n and returns the n-th Fibonacci sequence number (assume that the first 2 numbers are: 0, 1).

FUNCTIONS03-003: Write a recursive function that takes two numbers and returns their greatest common divisor (GCD) – най-голям общ делител. The greatest common divisor (GCD) of two numbers is the greatest common factor number that divides them without reminder (без остатък, точно).
For example: The GCD of 21 and 14 is 7.

FILES 01

FILES01-001: Working in the OS environment (use your own OS). Manualy create the following hierarchiacal directory structure:

./JSF-01/
./JSF-01/ARRAYS01/
./JSF-01/ARRAYS01/ARRAY01-008/
./JSF-01/ARRAYS01/ARRAY01-008/ARRAY01-008.js //empty file
./JSF-01/ARRAYS01/ARRAY01-008/ARRAY01-009.js //empty file
./JSF-01/ARRAYS01/ARRAY01-008/ARRAY01-010.js //empty file
./JSF-01/LOOPS01-010/
./JSF-01/LOOPS01-012/
./JSF-01/FILES01/FILES01-001.js
./JSF-01/FILES01/FILES01-002.js

Make sure that you have the rights to modify/edit the files.
For each of the following exercises, all tasks must be solved with functions.

1) Rename the file “ARRAY01-010.js” to “mod-ARRAY01-010.js”.

2) Delete the file “FILES01-002.js”.

3) Open file “FILES01-001.js” and display its contents on the console. Move the file into the following directory:

./JSF-01/ARRAYS01/

4) Use the method createWriteStream() to write English alphabet into file ./JSF-01/FILE01/FILES01-003.js

FILES01-002: A given file has the following structure:

every line is either empty or contains a path or is (can be absolute, can be relative)
every line can have leading white spaces
every line can have trailing white spaces
every path ends with "/" (backslash)
skip all lines, which do not comply with these rules

Example of such file:

./test/
./notes/
./JSF-01/ARRAYS01/ARRAY02/
./JSF-01/FUNCTIONS01/
./JSF-01/LOOPS02/
./JSF-01/FILES02/
/work/JSF/JSF-01/STRINGS09/
./JSF-01/FILES01/
/work/JSF-Par1/JSF/JSF-01/LOOPS01/

Read the file and create (with JavaScript fs methods) the directories.

Hints:

Check if a certain directory exists.
Use the following methods:
Node fs.mkdir()
Node fs.exists()