**FONTEND ADVANCED CODING CHALLENGES**

**JAVASCRIPT MODULE I**

**CODING CHALLENGE PART 1**

**Coding Challenge #1**

Jack and Jill are trying to compare their BMI (Body Mass Index), which is calculated using the formula:

BMI = mass / height \*\* 2 = mass / (height \* height) (mass in kg and height in meter).

**Your tasks:**

1. Store Jack's and Jill's mass and height in variables
2. Calculate both their BMIs using the formula (you can even implement both versions)
3. Create a Boolean variable 'jackHigherBMI' containing information about whether Jack has a higher BMI than Jill.

**Test data:**

* Data 1: Jacks weights 78 kg and is 1.69 m tall. Jill weights 92 kg and is 1.95m tall.
* Data 2: Jacks weights 95 kg and is 1.88 m tall. Jill weights 85 kg and is 1.76m tall.

**Coding Challenge #2**

Use the BMI example from Challenge #1, and the code you already wrote, and improve it.

**Your tasks:**

1. Print a nice output to the console, saying who has the higher BMI. The message is either *"Jack's BMI is higher than Jill's!"* or *"Jill's BMI is higher than Jack's!"*
2. Use a template literal to include the BMI values in the outputs. Example: *"Jack's BMI (28.3) is higher than Jill's (23.9)!"*

**Hint:** Use an if/else statement

**Coding Challenge #3**

There are two gymnastics teams, **Dolphins** and **Koalas**. They compete against each other 3 times. The winner with the highest average score wins a trophy!

**Your tasks:**

1. Calculate the average score for each team, using the test data below
2. Compare the team's average scores to determine the winner of the competition, and print it to the console. Don't forget that there can be a draw, so test for that as well (draw means they have the same average score)
3. Include a requirement for a minimum score of 100. With this rule, a team only wins if it has a higher score than the other team, and the same time a score of at least 100 points.
4. Minimum score also applies to a draw! So a draw only happens when both teams have the same score and both have a score greater or equal 100 points. Otherwise, no team wins the trophy

**Test data:**

* Data 1: Dolphins score 96, 108 and 89. Koalas score 88, 91 and 110
* Data Bonus 1: Dolphins score 97, 112 and 101. Koalas score 109, 95 and 123
* Data Bonus 2: Dolphins score 97, 112 and 101. Koalas score 109, 95 and 106

**Coding Challenge #4**

Steven wants to build a very simple tip calculator for whenever he goes eating in a

restaurant. In his country, it's usual to tip 15% if the bill value is between 50 and

300. If the value is different, the tip is 20%.

**Your tasks:**

1. Calculate the tip, depending on the bill value. Create a variable called 'tip' for this. It's not allowed to use an if/else statement (If it's easier for you, you can start with an if/else statement, and then try to convert it to a ternary operator!)
2. Print a string to the console containing the bill value, the tip, and the final value (bill + tip). Example: *“The bill was 275, the tip was 41.25, and the total value 316.25”*

**Test data:**

* Data 1: Test for bill values 275, 40 and 430

**Hints:**

* To calculate 20% of a value, simply multiply it by 20/100 = 0.2
* Value X is between 50 and 300, if it's >= 50 && <= 300

**CODING CHALLENGE PART 2**

**Coding Challenge #1**

Back to the two gymnastics teams, the Dolphins and the Koalas! There is a new gymnastics discipline, which works differently.

Each team competes 3 times, and then the average of the 3 scores is calculated (so one average score per team).

A team **only** wins if it has at least **double** the average score of the other team. Otherwise, no team wins!

**Your tasks:**

1. Create an arrow function 'calcAverage' to calculate the average of 3 scores
2. Use the function to calculate the average for both teams
3. Create a function 'checkWinner' that takes the average score of each team as parameters ('avgDolhins' and 'avgKoalas'), and then logs the winner to the console, together with the victory points, according to the rule above. Example: *"Koalas win (30 vs. 13)"*
4. Use the 'checkWinner' function to determine the winner for both Data 1 and Data 2
5. Ignore draws this time

**Test data:**

* Data 1: Dolphins score 44, 23 and 71. Koalas score 65, 54 and 49
* Data 2: Dolphins score 85, 54 and 41. Koalas score 23, 34 and 27

**Hints:**

To calculate average of 3 values, add them all together and divide by 3

To check if number A is at least double number B, check for A >= 2 \* B.

Apply this to the team's average scores

**Coding Challenge #2**

Steven is still building his tip calculator, using the same rules as before: Tip 15% of

the bill if the bill value is between 50 and 300, and if the value is different, the tip is

20%.

**Your tasks:**

1. Write a function 'calcTip' that takes any bill value as an input and returns the corresponding tip, calculated based on the rules above (you can check out the code from first tip calculator challenge if you need to). Use the function type you like the most. Test the function using a bill value of 100
2. And now let's use arrays! So create an array 'bills' containing the test data below
3. Create an array 'tips' containing the tip value for each bill, calculated from the function you created before
4. Create an array 'total' containing the total values, so the bill + tip

**Test data:** 125, 555 and 44

**Hint:** Remember that an array needs a value in each position, and that value can

actually be the returned value of a function! So you can just call a function as array

values (so don't store the tip values in separate variables first, but right in the new

array)

**Coding Challenge #3**

Let's go back to Jack and Jill comparing their BMIs! This time, let's use objects to

implement the calculations! Remember: BMI = mass / height \*\* 2 = mass

/ (height \* height) (mass in kg and height in meter)

**Your tasks:**

1. For each of them, create an object with properties for their full name, mass, and height (Jack Miller and Jill Smith)
2. Create a 'calcBMI' method on each object to calculate the BMI (the same method on both objects). Store the BMI value to a property, and also return it from the method
3. Log to the console who has the higher BMI, together with the full name and the respective BMI. Example: *"Jill's BMI (28.3) is higher than Jack's (23.9)!"*

**Test data:** Jacks weights 78 kg and is 1.69 m tall. Jill weights 92 kg and is 1.95 m

tall.

**Coding Challenge #4**

Let's improve Steven's tip calculator even more, this time using loops!

**Your tasks:**

1. Create an array 'bills' containing all 10 test bill values
2. Create empty arrays for the tips and the totals ('tips' and 'totals')
3. Use the 'calcTip' function we wrote before (no need to repeat) to calculate tips and total values (bill + tip) for every bill value in the bills array. Use a for loop to perform the 10 calculations!

**Test data:** 22, 295, 176, 440, 37, 105, 10, 1100, 86 and 52

**Hints:** Call ‘calcTip ‘in the loop and use the push method to add values to the tips and totals arrays

**Bonus:**

4. Write a function 'calcAverage' which takes an array called 'arr' as an argument. This function calculates the average of all numbers in the given array. This is a **difficult** challenge (you haven't done this before)! Here is how to solve it:

4.1. First, you will need to add up all values in the array. To do the addition, start by creating a variable 'sum' that starts at 0. Then loop over the array using a for loop. In each iteration, add the current value to the 'sum' variable. This way, by the end of the loop, you have all values added together

4.2. To calculate the average, divide the sum you calculated before by the length of the array (because that's the number of elements)

4.3. Call the function with the 'totals' array

**CODING CHALLENGE PART 3**

**Coding Challenge #1**

Given an array of forecasted maximum temperatures, the thermometer displays a

string with the given temperatures. **Example:** [17, 21, 23] will print *"... 17ºC in 1*

*days ... 21ºC in 2 days ... 23ºC in 3 days ..."*

**Your tasks:**

1. Create a function 'printForecast' which takes in an array 'arr' and logs a string like the above to the console. Try it with both test datasets.
2. Use the problem-solving framework: Understand the problem and break it up into sub-problems!

**Test data:**

* Data 1: [17, 21, 23]
* Data 2: [12, 5, -5, 0, 4]

**FUNCTIONS CODING CHALLENGE**

**Coding Challenge #1**

Let's build a simple poll app!

A poll has a question, an array of options from which people can choose, and an array with the number of replies for each option. This data is stored in the starter 'poll' object below.

**Your tasks:**

1. Create a method called 'registerNewAnswer' on the 'poll' object. The method does 2 things:

1.1. Display a prompt window for the user to input the number of the selected option. The prompt should look like this:

*What is your favourite programming language?*

*0: JavaScript*

*1: Python*

*2: Rust*

*3: C++*

*(Write option number)*

1.2. Based on the input number, update the 'answers' array property. For example, if the option is 3, increase the value **at position** 3 of the array by 1. Make sure to check if the input is a number and if the number makes sense (e.g. answer 52 wouldn't make sense, right?)

1. Call this method whenever the user clicks the *"Answer poll"* button.
2. Create a method 'displayResults' which displays the poll results. The method takes a string as an input (called 'type'), which can be either *'string'* or *'array'*. If type is *'array'*, simply display the results array as it is, using console.log(). This should be the default option. If type is *'string'*, display a string like *"Poll results are 13, 2, 4, 1".*
3. Run the 'displayResults' method at the end of each 'registerNewAnswer' method call.
4. **Bonus:** Use the 'displayResults' method to display the 2 arrays in the test data. Use both the *'array'* and the *'string'* option. Do **not** put the arrays in the poll object! So what should the this keyword look like in this situation?

**Test data for bonus:**

* Data 1: [5, 2, 3]
* Data 2: [1, 5, 3, 9, 6, 1]

*var* poll = {

question: "What is your favourite programming language?",

options: ["0: JavaScript", "1: Python", "2: Rust", "3: C++"],

answers: new Array(4).fill(0),

};

**Coding Challenge #2**

This is more of a *thinking* challenge than a *coding* challenge

**Your tasks:**

1. Take the IIFE below and at the end of the function, attach an event listener that changes the color of the selected h1 element ('header') to blue, each time the body element is clicked. Do **not** select the h1 element again!
2. And now explain to **yourself** (or someone around you) **why** this worked! Take all the time you need. Think about **when** exactly the callback function is executed, and what that means for the variables involved in this example.

(*function* () {

*const* header = document.querySelector('h1');

header.style.color = 'red';

})();

**Working With Arrays**

**Coding Challenge #1**

Julia and Kate are doing a study on dogs. So each of them asked 5 dog owners about their dog's age, and stored the data into an array (one array for each). For now, they are just interested in knowing whether a dog is an adult or a puppy. A dog is an adult if it is at least 3 years old, and it's a puppy if it's less than 3 years old.

**Your tasks:**

Create a function 'checkDogs', which accepts 2 arrays of dog's ages ('dogsJulia' and 'dogsKate'), and does the following things:

1. Julia found out that the owners of the **first** and the **last two** dogs actually have cats, not dogs! So create a shallow copy of Julia's array, and remove the cat ages from that copied array (because it's a bad practice to mutate function parameters)
2. Create an array with both Julia's (corrected) and Kate's data
3. For each remaining dog, log to the console whether it's an adult (*"Dog number 1 is an adult, and is 5 years old"*) or a puppy (*"Dog number 2 is still a puppy"*)
4. Run the function for both test datasets

**Test data:**

* Data 1: Julia's data [3, 5, 2, 12, 7], Kate's data [4, 1, 15, 8, 3]
* Data 2: Julia's data [9, 16, 6, 8, 3], Kate's data [10, 5, 6, 1, 4]

**Coding Challenge #2**

Let's go back to Julia and Kate's study about dogs. This time, they want to convert dog ages to human ages and calculate the average age of the dogs in their study.

**Your tasks:**

Create a function 'calcAverageHumanAge', which accepts an arrays of dog's ages ('ages'), and does the following things in order:

1. Calculate the dog age in human years using the following formula: if the dog is <= 2 years old, humanAge = 2 \* dogAge. If the dog is > 2 years old, humanAge = 16 + dogAge \* 4
2. Exclude all dogs that are less than 18 human years old (which is the same as keeping dogs that are at least 18 years old)
3. Calculate the average human age of all adult dogs (you should already know from other challenges how we calculate averages)
4. Run the function for both test datasets

**Test data:**

* Data 1: [5, 2, 4, 1, 15, 8, 3]
* Data 2: [16, 6, 10, 5, 6, 1, 4]

Rewrite the 'calcAverageHumanAge' function from Challenge #2, but this time

as an arrow function, and using chaining!

**Test data:**

* Data 1: [5, 2, 4, 1, 15, 8, 3]
* Data 2: [16, 6, 10, 5, 6, 1, 4]

**Coding Challenge #4**

Julia and Kate are still studying dogs, and this time they are studying if dogs are eating too much or too little.

Eating too much means the dog's current food portion is larger than the recommended portion, and eating too little is the opposite.

Eating an okay amount means the dog's current food portion is within a range 10% above and 10% below the recommended portion (see hint).

**Your tasks:**

1. Loop over the 'dogs' array containing dog objects, and for each dog, calculate the recommended food portion and add it to the object as a new property. Do **not** create a new array, simply loop over the array. Forumla:
2. recommendedFood = weight \*\* 0.75 \* 28. (The result is in grams of food, and the weight needs to be in kg)
3. Find Sarah's dog and log to the console whether it's eating too much or too little. **Hint:** Some dogs have multiple owners, so you first need to find Sarah in the owners array, and so this one is a bit tricky (on purpose)
4. Create an array containing all owners of dogs who eat too much ('ownersEatTooMuch') and an array with all owners of dogs who eat too little ('ownersEatTooLittle').
5. Log a string to the console for each array created in 3., like this: *"Matilda and Alice and Bob's dogs eat too much!"* and *"Sarah and Jill and Michael's dogs eat too little!"*
6. Log to the console whether there is any dog eating **exactly** the amount of food that is recommended (just true or false)
7. Log to the console whether there is any dog eating an **okay** amount of food (just true or false)
8. Create an array containing the dogs that are eating an **okay** amount of food (try to reuse the condition used in 6.)
9. Create a shallow copy of the 'dogs' array and sort it by recommended food portion in an ascending order (keep in mind that the portions are inside the array's objects

**Hints:**

Being within a range 10% above and below the recommended portion means:

current > (recommended \* 0.90) && current < (recommended \* 1.10)

Basically, the current portion should be between 90% and 110% of the recommended portion.

*var* dogs = [

{ weight: 22, curFood: 250, owners: ['Alice', 'Bob'] },

{ weight: 8, curFood: 200, owners: ['Matilda'] },

{ weight: 13, curFood: 275, owners: ['Sarah', 'Jill'] },

{ weight: 32, curFood: 340, owners: ['Michael'] },

];