**[IS216] Extra Exercises - Week 4 - JavaScript Syntax & Operations**

**Instructions (G4 & G5):**

* **You are required to attempt Q1-Q8, the remaining questions are optional.**
* **Submit 1 single file (js-extras.html) with the code inside for Q1-8 to eLearn by 8:15am next Wed.**

**Objectives**

● To get familiar with JavaScript syntaxes

● To practice on applying JavaScript operations to solve programming problems

**Instructions**

● Questions with no asterisk mark are easy.

● Questions marked with \* are slightly challenging.

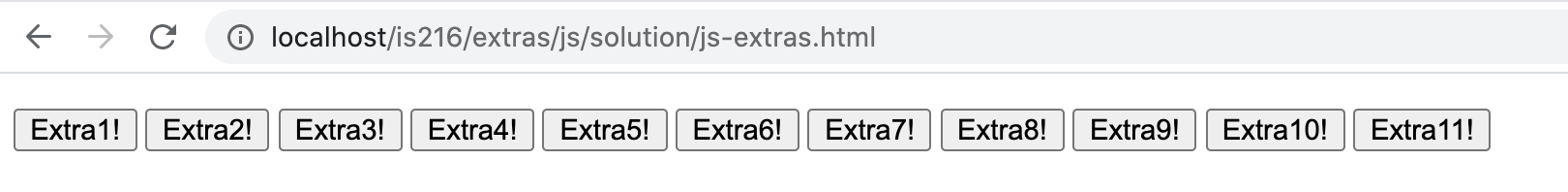
● Questions marked with **\***\* are challenging.

● Questions marked with \*\*\* are very challenging.

**Required**

* **Resources** folder

**All the following exercises can be done in js-extras.html in resource folder**



**Question 1 (\*)**

Write a function that computes and prints out the area and circumference of a circle of a given radius.

The radius could be set to an int or float value. You can set the value of π to 3.14

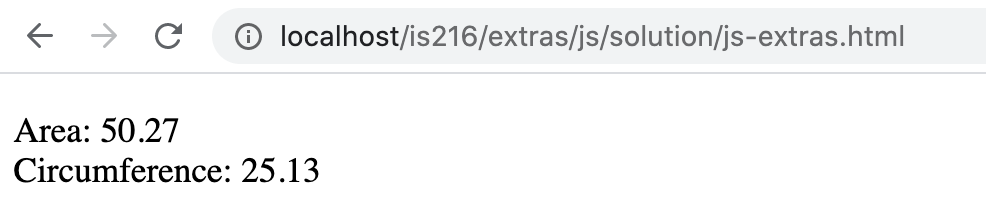
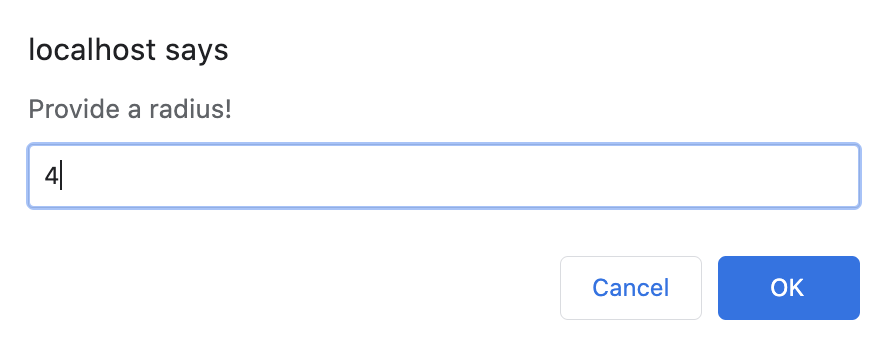
Formulas:

· Area of circle = π X radius2

· Circumference of circle = π X 2 X radius

Hint: explore Math.PI()

See examples below:

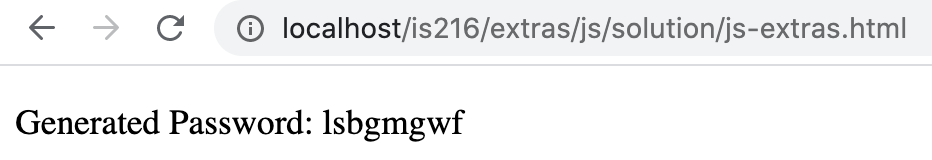


**Question 2 (\*\*)**

Write a function that generates a random password of 8 characters.

For example, password -> cxqaptgh

Hint: explore Math.random(), Math.floor()



**Question 3 (\*)**

In number theory, a perfect number is a positive integer that is equal to the sum of all its factors excluding itself.

For example, 6 is a perfect number because the sum of its factors i.e. 1 + 2 + 3 = 6

Write a function that takes in a positive integer. The function should then check if the number entered is a perfect number or not and print the result. You can assume that only positive integers are entered.

Some examples of perfect numbers are 6, 28, 496

|  |  |
| --- | --- |
| The user enters number ‘4’ | Number ‘4’ is not a perfect number: |
| The user enters number ‘6’ | Number ‘6’ is a perfect number: |

**Question 4 (\*\*)**

Write a function that converts a given integer into its equivalent binary number.

For example, 10 -> 1010, 12 -> 1100

Your function must validate that the user enters a valid integer. That is, ask the user to enter an integer again if she/he enters a non-integer value, e.g. ‘abc’

Hint: explore Number(), Number.isInteger(), toString()

|  |  |
| --- | --- |
| User enters a non-integer value ‘abc’ | The program prompts the user to enter a valid integer: |
| User enters a decimal value ‘2.4’ | The program prompts the user to enter a valid integer: |
| User enters an integer ‘68’ | Its equivalent binary number: |

**Question 5 (\*\*)**

Write a function that requests the user’s name so that it displays back with the surname in capital letters and the rest of the name with first letter in uppercase.

Assume the following:

a) Two adjacent words of the name is separated by 1 space

b) If the name has 1 word, it is the surname

c) If the name has 2 words, the second word is the surname

d) If the name has more than 2 words, the first word is always the surname

For example, tan -> TAN, tan wee kiat -> TAN Wee Kiat, mary lim -> LIM Mary

Hint: use toUpperCase(), charAt()

|  |  |
| --- | --- |
| The name that has 1 word | The result |
| The name that has 2 words | The result |
| The name that has 3 words | The result |

**Question 6 (\*\*)**

Write a function that asks the user to enter a string, say txt, and two characters, say st and en.

The function then searches for a substring in txt such that the substring begins with the character st and ends with the character en.

If there is no such substring, display the message "No such substring".

Assume that the search is case-sensitive and only search for the first occurrence of the substring

Hint: explore indexOf, search(), includes()

For example,

txt = Fibonacci series

st = F

en = b

==> Substring [Fib] is found

txt = Fibonacci series

st = b

en = y

==> No such substring

txt = Fibonacci series

st = F

en = F

==> Substring [F] is found

**Question 7 (\*\*)**

Write a function that asks the user to enter two strings, say str1 and str2.

The function displays "Bingo!" if every character in str1 also appears in str2. Otherwise, it displays "Nope :("

Hint: explore includes()

For example,

str1 = daily

str2 = Holiday

=> Bingo!

str1 = lily

str2 = Holiday

=> Bingo!

str1 = pokemon

str2 = pogo

=> Nope :(

**Question 8 (\*\*)**

Write a function that requests two integers, say min and max. The function then displays all the perfect squares between min and max, inclusive.

For example, given min=10 and max=110, it displays the perfect squares – 16 25 36 49 64 81 100.

Your function must validate that the user provides only numbers (decimal numbers, e.g., 10.2, are accepted).

Hint: explore Math.sqrt(), Math.floor(), Math.ceil(), Math.pow()

|  |  |
| --- | --- |
| The user enters an invalid number | The result |
| The user enters a min number ‘9’ and a max number ‘109’ | The result |
| The user enters a min decimal number ‘10.2’ and a max number ‘100’ | The result |

**Question 9 (\*\*\*)**

Write a function that gets a set of integer inputs from the user. Assume that the user enters each integer, separated by a space. For example, 6 12 4 10

Your function validates that the user enters valid integers.

It should then display the minimum, maximum and median of all numbers entered.

Note: The median is the middle of the list of numbers.

For example, the median of numbers 12, 4, 5 is 5. In case of odd amount of numbers, the median is the exact middle number of numbers when arranged sorted.

In case of even amount of numbers, we would get a pair of middle numbers.

The median is half way between them.

For example,

median of numbers 6, 12, 4, 10 is 8 (6 + 10) / 2

because when placed in order the middle numbers would be 6 and 10.

Hint: explore split(), sort()

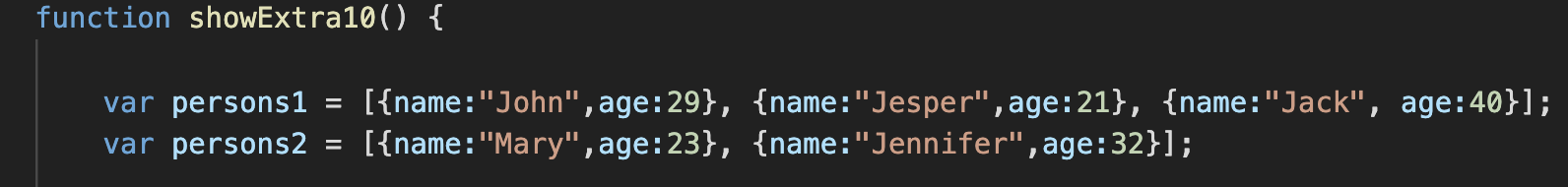
|  |  |
| --- | --- |
| The user enters 4 valid integers: | The result: |
| The user enters 7 valid integers: | The result: |
| The user enters an invalid integer: | The result: |

**Question 10 (\*\*\*)**

Write a function that merges two arrays of objects, representing persons.

Each person object has name and age properties. Assume that names are unique.

Sample arrays of objects are given in **js-extras.html**

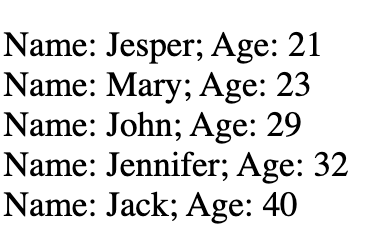


The function merges the two arrays into a single array in which the persons are ordered by their age.

Also create your own arrays of objects in the function to test your code.

Hint: explore concat()

**Example:**



**Question 11 (\*\*)**

Write a function that computes the total amount to be paid, based on an array of purchase items. A purchase item is an object, having **name** and **price** properties.

For example, name="pencil", price=1.25

Create your own arrays of objects in the function.