Help to complete the tasks of this exercise can be found on the chapter 8 ”Bugs and Errors” and chapter 9 “Regular Expressions” of our course book “Eloquent JavaScript” (3rd edition) by Marijin Haverbeke. The aims of the exercise are to learn to handle exceptional error situations and to use regular expressions to validate and modify strings.

Embed your theory answers, drawings, codes, and screenshots directly into this document. Always immediately after the relevant question. Return the document into your return box in itsLearning by the deadline.

Remember to give your own assessment when returning this document.

It’s also recommendable to use Internet sources to supplement the information provided by the course book.

The maximum number of points you can earn from this exercise is 10 + 1 = 11.

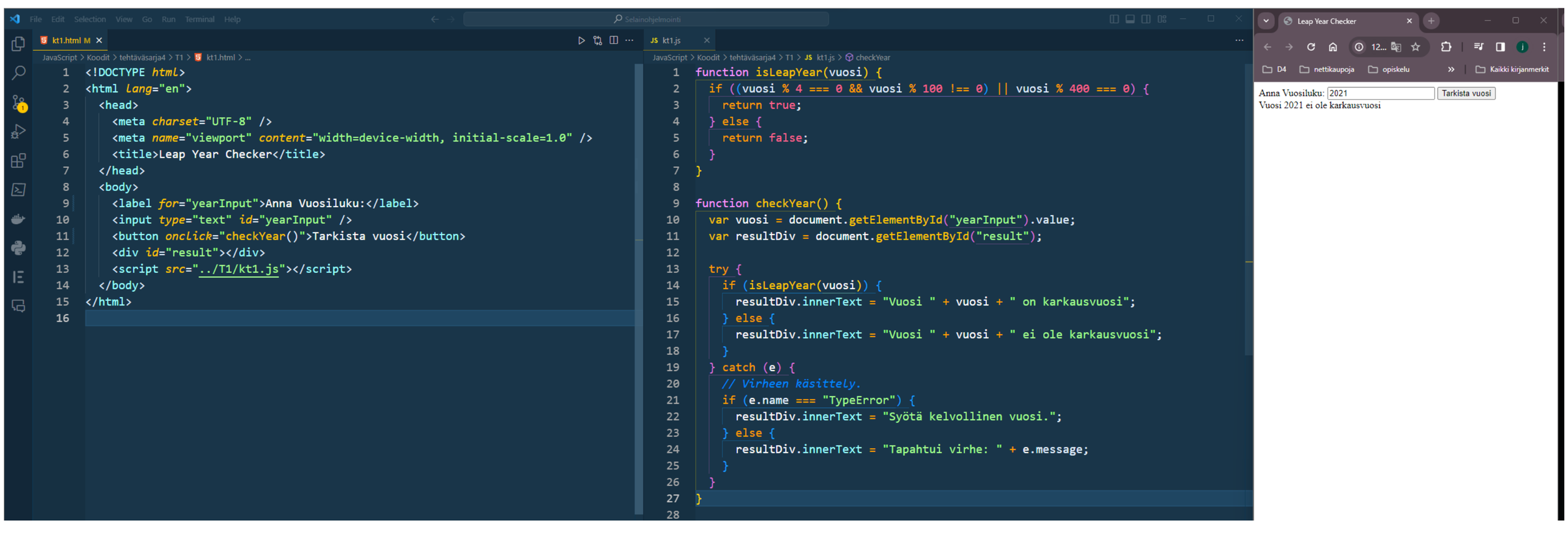
Tasks:

1. Adding error handling into a function. (1 point)

In the practical exercise 2 the task 1 was to program a function isLeapYear. The function accepted one argument year.

Develop this function further so that it can throw two different kinds of errors when called incorrectly. If the function is called without a parameter, then an error with a message “Missing argument year error” is thrown. If the function is called with non-integer argument, then an error with a message “Non-integer argument year error” is thrown. In this case, write the try-catch statement inside the function and output the error message on the console.

Please note that an argument value 0 should not be considered as an error.



2. Catching the error outside the function. (1 point)

Return to the task above. Change the implementation so that the try-catch is moved to an enclosing operation tryIsLeapYear. The errors are still thrown inside the isLeapYear operation.

3. Throwing and catching the error outside the function. (1 point)

Return to the task above and change the implementation so that also throwing the error is done in the enclosing operation tryIsLeapYear.

4. Error handling best practices. (2 points)

a. In your operations logic, when should you use if statements and when error handling? Give a justified explanation. (0,5 points)

b. Use Internet and write yourself a short list of the best practices of JavaScript error handling. (You don’t need elaborate on asynchronous code yet.) (1 point)

c. Analyze the error handling solutions of the above tasks. What are the benefits and drawbacks? What do the best practices say about them? (0,5 points)

5. What are the risks and benefits of error handling in general? Consider this carefully, because it is very important. (1 point)

6. Justify the existence of regular expressions. Why do we need them? Any drawbacks? (1 point)

7. Understanding the functioning of regular expressions. (4 * 0,5 = 2 points)

Explain shortly:

a. How regular expressions consume their input? (What is the mechanism of matching?)

b. What do parentheses around any part of the regular expression cause?

c. What is the difference between lazy matching and greedy matching in regular expressions?

d. When do you need to use RegExp constructor like new RegExp() instead of regular expression literal like / /?

8. Using regular expressions. (2 points)

Develop a function buildRegisterNumber so that it can throw two different kinds of errors when called with incorrect argument values. The function takes two arguments: theLetters and theDigits. If the value of theLetters argument is not valid, then an error with a message “Invalid register number letters” is thrown. If the value of theDigits argument is not valid, then an error with a message “Invalid register number digits” is thrown. In case the arguments are valid, then a valid register number is returned.

Use regular expressions to validate the argument values.

Let’s agree that a valid register number obeys the following rules

- there are from two to three uppercase letters before a dash
- the letter W is not allowed
- a dash
- there are from one to three digits after the dash
- no leading zeros are allowed (no zeros before first non-zero digit)

Examples of valid register numbers:

AX-12

UUI-6

GFS-200

Examples of invalid register numbers:

X-100

YUT-020

WWW-100