

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.

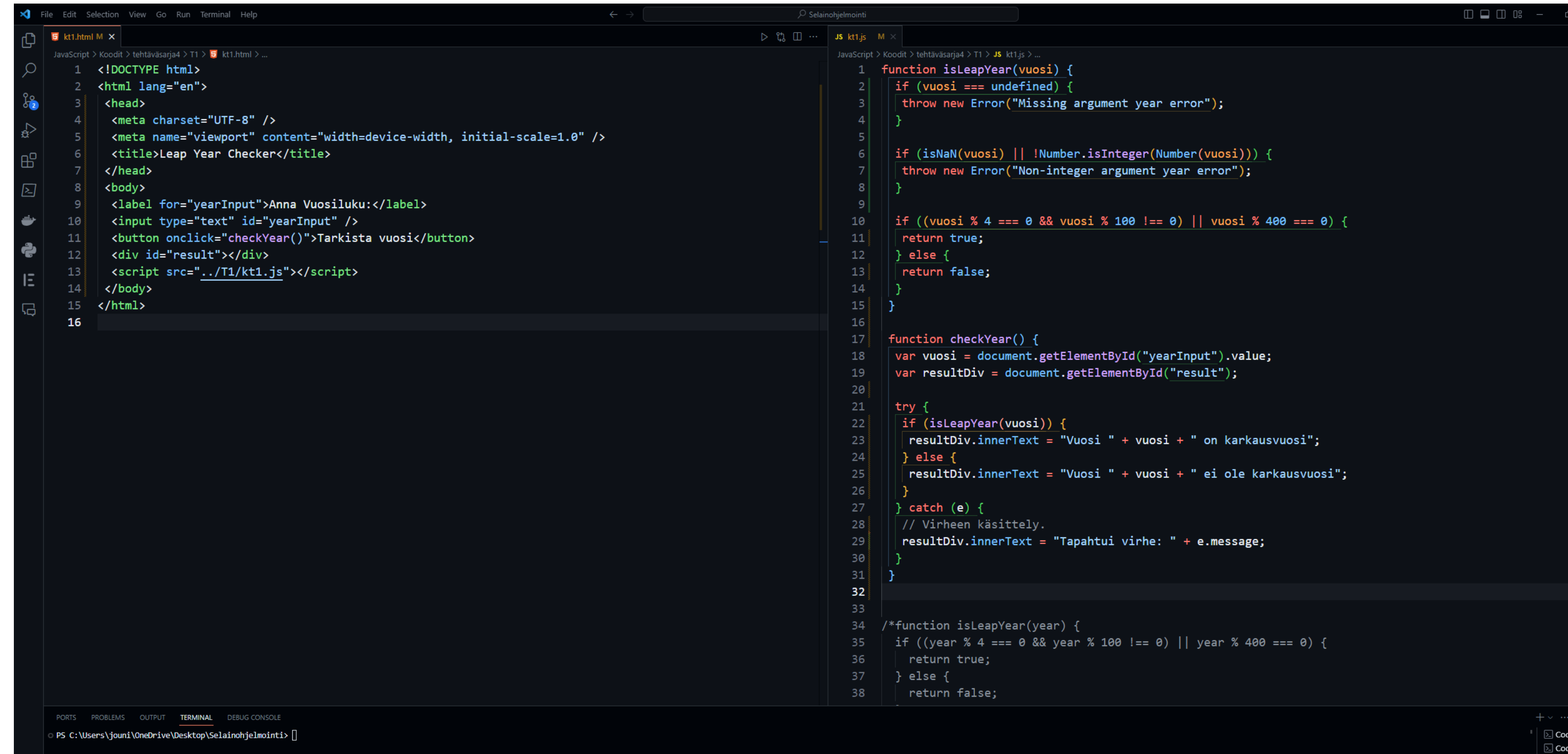
Tasks:

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.

The image shows a VS Code editor with two files open: `kt2.html` and `kt2.js`.

kt2.html (Left Panel):

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <title>Leap Year Checker</title>
7   </head>
8   <body>
9     <label for="yearInput">Anna Vuosiluku:</label>
10    <input type="text" id="yearInput" />
11    <button onclick="checkYear()">Tarkista vuosi</button>
12    <div id="result"></div>
13    <script src="kt2.js"></script>
14  </body>
15 </html>
16
```

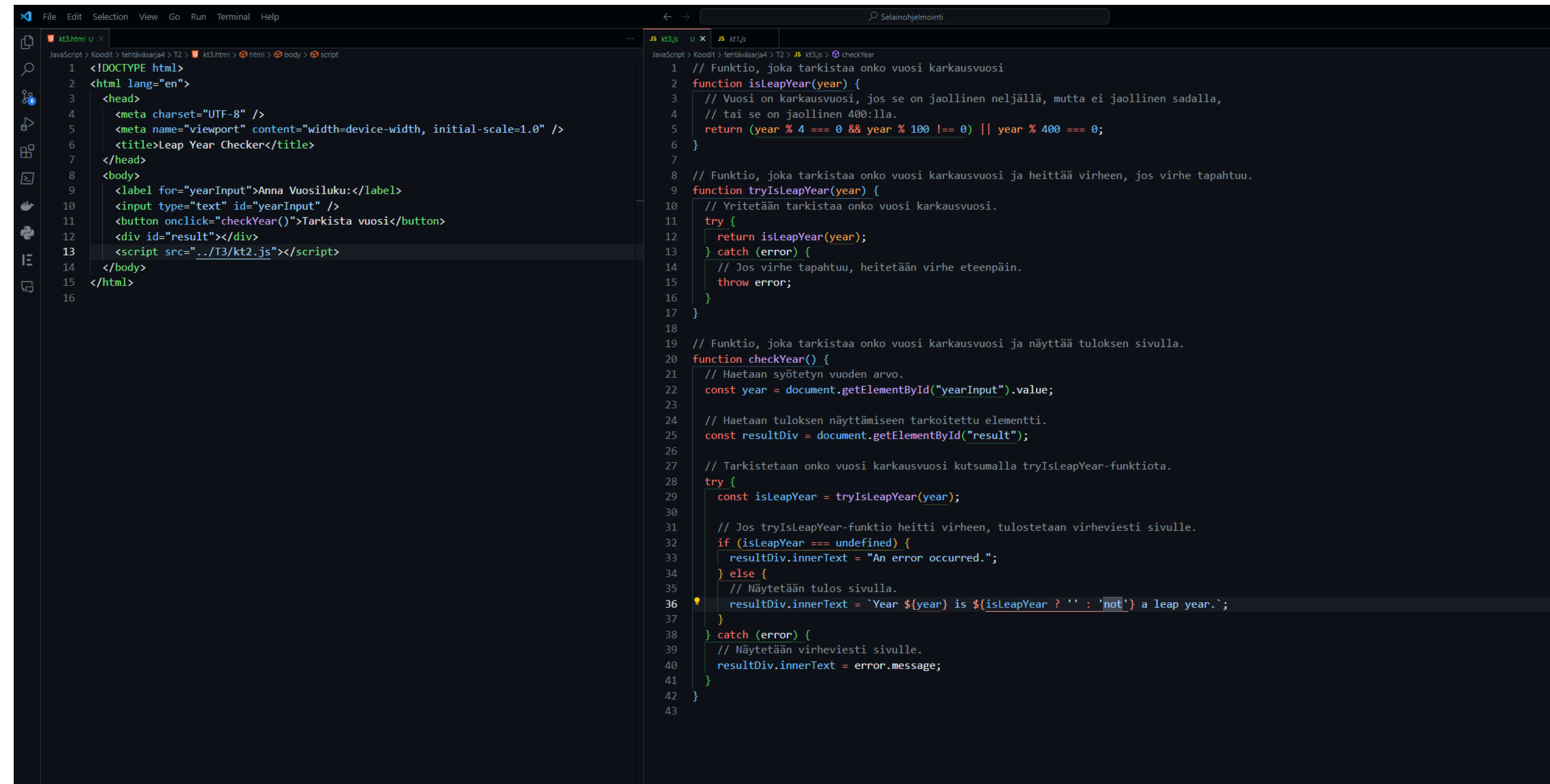
kt2.js (Right Panel):

```
1 // Funktio, joka tarkistaa onko vuosi karkausvuosi
2 function isLeapYear(vuosi) {
3   // Vuosi on karkausvuosi, jos se on jaollinen neljällä, mutta ei jaollinen sadalla,
4   // tai se on jaollinen 400:lla.
5   return (vuosi % 4 === 0 && vuosi % 100 !== 0) || vuosi % 400 === 0;
6 }
7
8
9 // Tein tähän kohtaan tämän Catching the error outside the function
10 // Funktio, joka yrittää tarkistaa onko vuosi karkausvuosi ja heittää virheen, jos virhe tapahtuu.
11 function tryIsLeapYear(vuosi) {
12   // Yritetään tarkistaa onko vuosi karkausvuosi.
13   try {
14     return isLeapYear(vuosi);
15   } catch (virhe) {
16     // Jos virhe tapahtuu, heitetään virhe eteenpäin.
17     throw virhe;
18   }
19 }
20
21 // Funktio, joka tarkistaa onko vuosi karkausvuosi ja näyttää tuloksen sivulla.
22 function checkYear() {
23   // Haetaan syötetyn vuoden arvo.
24   const vuosi = document.getElementById("yearInput").value;
25
26   // Haetaan tuloksen näyttämiseen tarkoitettu elementti.
27   const resultDiv = document.getElementById("result");
28
29   // Tarkistetaan onko vuosi karkausvuosi kutsumalla tryIsLeapYear-funktiota.
30   try {
31     const onKarkausvuosi = tryIsLeapYear(vuosi);
32
33     // Jos tryIsLeapYear-funktio heitti virheen, tulostetaan virheviesti sivulle.
34     if (onKarkausvuosi === undefined) {
35       resultDiv.innerText = "Tapahtui virhe.";
36     } else {
37       // Näytetään tulos sivulla.
38       resultDiv.innerText = `Vuosi ${vuosi} on ${onKarkausvuosi ? '' : 'ei '}karkausvuosi.`;
39     }
40   } catch (virhe) {
41     // Näytetään virheviesti sivulle.
42     resultDiv.innerText = virhe.message;
43   }
44 }
45
```

The bottom status bar shows the file path: `PS C:\Users\jouni\OneDrive\Desktop\Selainohjelmointi>`.

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.



```
JavaScript > Koodit > tehtäväsarja4 > T2 > kt3.html > html > body > script
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <title>Leap Year Checker</title>
7   </head>
8   <body>
9     <label for="yearInput">Anna Vuosiluku:</label>
10    <input type="text" id="yearInput" />
11    <button onclick="checkYear()">Tarkista vuosi</button>
12    <div id="result"></div>
13    <script src="../../T3/kt2.js"></script>
14  </body>
15 </html>
16

JavaScript > Koodit > tehtäväsarja4 > T2 > JS kt3.js > checkYear
1 // Funktio, joka tarkistaa onko vuosi karkausvuosi
2 function isLeapYear(year) {
3   // Vuosi on karkausvuosi, jos se on jaollinen neljällä, mutta ei jaollinen sadalla,
4   // tai se on jaollinen 400:lla.
5   return (year % 4 === 0 && year % 100 !== 0) || year % 400 === 0;
6 }
7
8 // Funktio, joka tarkistaa onko vuosi karkausvuosi ja heittää virheen, jos virhe tapahtuu.
9 function tryIsLeapYear(year) {
10  // Yritetään tarkistaa onko vuosi karkausvuosi.
11  try {
12    return isLeapYear(year);
13  } catch (error) {
14    // Jos virhe tapahtuu, heitetään virhe eteenpäin.
15    throw error;
16  }
17 }
18
19 // Funktio, joka tarkistaa onko vuosi karkausvuosi ja näyttää tuloksen sivulla.
20 function checkYear() {
21  // Haetaan syötetyn vuoden arvo.
22  const year = document.getElementById("yearInput").value;
23
24  // Haetaan tuloksen näyttämiseen tarkoitettu elementti.
25  const resultDiv = document.getElementById("result");
26
27  // Tarkistetaan onko vuosi karkausvuosi kutsumalla tryIsLeapYear-funktiota.
28  try {
29    const isLeapYear = tryIsLeapYear(year);
30
31    // Jos tryIsLeapYear-funktio heitti virheen, tulostetaan virheviesti sivulle.
32    if (isLeapYear === undefined) {
33      resultDiv.innerText = "An error occurred.";
34    } else {
35      // Näytetään tulos sivulla.
36      resultDiv.innerText = `Year ${year} is ${isLeapYear ? '' : 'not'} a leap year.`;
37    }
38  } catch (error) {
39    // Näytetään virheviesti sivulle.
40    resultDiv.innerText = error.message;
41  }
42 }
43
```

4. Error handling best practices. (2 points)

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

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

Säännölliset lausekkeet (regular expressions) käyvät läpi syötteensä tietyn säännön mukaisesti. Ne vertaavat säännöllistä lauseketta syötteen eri osiin ja yrittävät löytää vastaavuutta. Matching-mekanismi tarkoittaa sitä prosessia, jossa säännöllinen lauseke vertaillaan syötteen kanssa, ja jos se löytää vastaavuuden, se ilmoitetaan tai tallennetaan. Tämä mahdollistaa esimerkiksi tietyn kuvion tai merkkijonon löytämisen suuremmasta tekstimassasta.

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

Sulkeet säännöllisen lausekkeen osan ympärillä merkitsevät ryhmää, mikä vaikuttaa siihen, miten lauseke sovitetaan syötteeseen ja miten siihen viitataan myöhemmin. Ryhmän käyttö mahdollistaa esimerkiksi tietyn osan merkityksen yhdistämisen tai toistamisen.



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

Laiska (lazy) ja ahne (greedy) sovitus säännöllisissä lausekkeissa viittaavat siihen, miten lauseke yrittää sovittaa syötettä.

- **Ahne sovitus (greedy matching):** Yrittää sovittaa mahdollisimman pitkän osan syötteestä, mikäli se täyttää lausekkeen ehdot.
- **Laiska sovitus (lazy matching):** Yrittää sovittaa mahdollisimman lyhyen osan syötteestä, mikäli se täyttää lausekkeen ehdot.

Esimerkiksi, jos käyttää lauseketta **a.*b** ahneesti, se voisi sovittaa koko välillä olevan osan, joka alkaa 'a' ja päättyy 'b'. Laiskassa sovituksessa se yrittäisi löytää mahdollisimman pienen osan, joka täyttää ehdot.

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

RegExp-konstruktori (new RegExp()) tarvitaan silloin, kun säännöllisen lausekkeen arvo ei ole tiedossa ennen ohjelman suoritusta, vaan se luodaan dynaamisesti. Toisin sanoen, jos haluat rakentaa säännöllisen lausekkeen merkkijonosta tai muista muuttujista ohjelman aikana, käytät RegExp-konstruktoriä.

Esimerkki:

```
var patternString = "ab+c";
```

```
var regexLiteral = /ab+c/; // Tämä on säännöllinen lauseke kirjoitettuna suoraan koodiin
```

```
var regexConstructor = new RegExp(patternString); // Tämä luo säännöllisen lausekkeen dynaamisesti
```

Jos säännöllisen lausekkeen arvo tiedetään jo koodin kirjoitusvaiheessa, voit käyttää suoraa säännöllisen lausekkeen kirjallista muotoa (**/ /**). Tämä on yleensä suositeltavaa, kun mahdollista, koska se on lyhyempi ja luettavampi.

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

FileEditSelectionViewGoRunTerminalHelp

Selainohjelmointi

JS k8.jsU x

JavaScript > Koodit > tehtävasarja4 > T8 > JS k8.js > ...

1function rakennaRekisterinumero(theLetters, theDigits) {
2// Määritä säännölliset lausekkeet validointia varten
3const lettersRegex = /^[A-VX-Z]{2,3}\$/; // kaksi tai kolme iso kirjainta, W ei ole sallittu
4const digitsRegex = /^[1-9]\d{0,2}\$/; // yhdestä kolmeen numeroa ilman johtavia nollia
5
6// Validoi theLetters-argumentti
7if (!lettersRegex.test(theLetters)) {
8| throw new Error("Virheelliset rekisterinumeron kirjaimet");
9| }
10
11// Validoi theDigits-argumentti
12if (!digitsRegex.test(theDigits)) {
13| throw new Error("Virheelliset rekisterinumeron numerot");
14| }
15
16// Jos molemmat argumentit ovat valideja, muodosta ja palauta rekisterinumero
17return `\${theLetters}-\${theDigits}`;
18}
19
20// Esimerkit
21try {
22| const validiRekisterinumero = rakennaRekisterinumero("AX", "12");
23| console.log("Validi rekisterinumero:", validiRekisterinumero);
24| } catch (error) {
25| console.error("Virhe:", error.message);
26| }
27
28try {
29| const virheellinenRekisterinumero = rakennaRekisterinumero("X", "100");
30| console.log("Validi rekisterinumero:", virheellinenRekisterinumero);
31| } catch (error) {

PORTSPROBLEMSOUTPUTTERMINALDEBUG CONSOLE

● PS C:\Users\jouni\OneDrive\Desktop\Selainohjelmointi> node "c:\Users\jouni\OneDrive\Desktop\Selainohjelmointi\JavaScript\Koodit\tehtävasarja4\T8\k8.js"

Validi rekisterinumero: AX-12

Virhe: Virheelliset rekisterinumeron kirjaimet

○ PS C:\Users\jouni\OneDrive\Desktop\Selainohjelmointi>

TURKU AMK

TURKU UNIVERSITY OF APPLIED SCIENCES