

**Lab 12 Javascript 5 – Loops and Function**

**Not to be redistributed  
to Course Hero or any  
other public websites**

**General Information****What you should do**

- You should first review Lecture 4 (Part B) and be familiar with the concepts on Javascript as well as the related code examples so that you can refer to the specific techniques from the corresponding slides when completing the tasks in this lab.
- You should try to come up with the code yourself as much as possible. Do not be afraid of making mistakes, since debugging (finding out where your code goes wrong and fixing it) is part of the learning process.
- We do not give out model programs to the exercises. There can be multiple ways to write the code that solves the same problem. It is important that you build up the program logic yourself instead of merely looking at some code that you do not understand. At any time if you are lost or if you have any questions, feel free to ask the instructor, tutor, or teaching assistant and we will be very happy to help you.

**Self-Discovery**

- Most lab tasks are designed to be relatively simple such that you can take the time to think about the related underlying concepts. Besides, we also encourage you to discover things on your own which may not be specified in the tasks.

**Introduction**

In this lab, we will work out a program related to the following problem that can be found on the web. Take a look at the problem first and see whether you are able to solve it.

This problem can be solved by pre-school children in five to ten minutes, by programmers in an hour and by people with higher education... well, check it yourself!

8809 = 6	5555 = 0
7111 = 0	8193 = 3
2172 = 0	8096 = 5
6666 = 4	1012 = 1
1111 = 0	7777 = 0
3213 = 0	9999 = 4
7662 = 2	7756 = 1
9313 = 1	6855 = 3
0000 = 4	9881 = 5
2222 = 0	5531 = 0
3333 = 0	2581 = ???

Reference: <https://www.theguardian.com/science/2015/jun/08/did-you-solve-it-are-you-smarter-than-a-hong-kong-six-year-old>

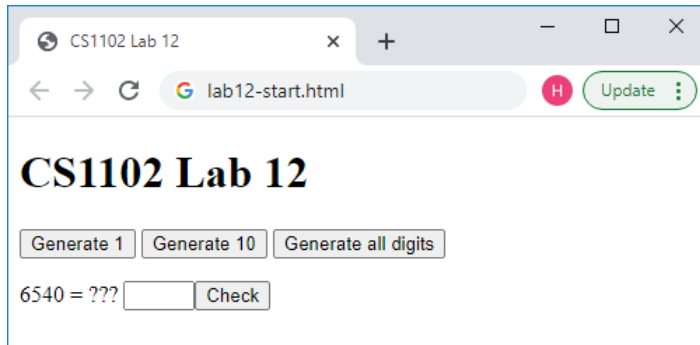
In this lab you will write a Javascript program to solve this kind of problem (thus demonstrating that as a programmer you can solve it in an hour).


**Important:** some sample lines of code are provided in this PDF file. **DO NOT directly copy and paste code from the PDF file to Komodo Edit. You should re-type the code in Komodo Edit instead.** Otherwise your code may not work correctly, especially for quotation marks ""

## Task 1.1 Download and examine the start code

In this lab you are given a webpage with some starting code. The webpage is stored in the file `lab12-start.html` which can be downloaded from the Canvas course page. Download this file and open it in Komodo Edit.

Also view this webpage on a browser.



Refresh the webpage a few times (by clicking the refresh button  or pressing F5 key) to see that each time it is generating 4 digits randomly in the last line. In Komodo Edit, examine the Javascript code in the function `init()` and `generateRandom4DigitText()` to understand how it works.

```
var clueQuestion;

function init(){

    clueQuestion = generateRandom4DigitText();

    document.getElementById("question").innerHTML = clueQuestion + " = ???";

}

function generateRandom4DigitText(){

    var i, digit, result;

    result = "";
    for (i=0; i<4; i++) {
        digit = Math.floor(Math.random()*10);
        result = result + digit;
    }

    return result;

}
```

This variable is declared outside any functions. A variable declared this way is a global variable which can be read and modified anywhere in the Javascript code

This variable is assigned a value returned after calling another function `generateRandom4DigitText()`

The HTML span tag with id `question` is updated with the text content specified as the expression on the right hand side

This variable is initialized as an empty string

`Math.random()` generates a random floating-point number from 0 (inclusive) to 1 (exclusive)  
`Math.floor()` is the floor function  
 The expression will thus generate a single number ranging from 0 to 9

Since `result` is a string, the expression will concatenate the digit to the end of this string

The value of the string `result` is returned

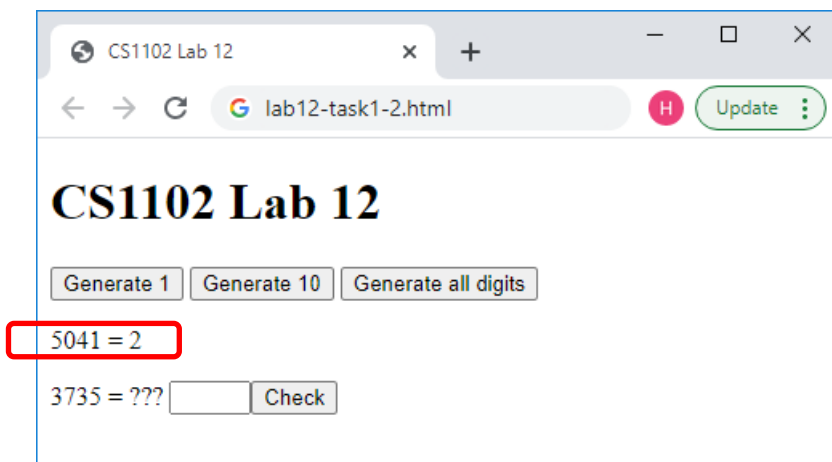
In addition to the above explanations, note that:

- The function `init()` is called when the webpage has finished loading (as indicated in the HTML code `<body onload="init();">`)
- The purpose of the function `generateRandom4DigitText()` is to generate 4 random digits. The for-loop inside this function runs with 4 iterations where in each iteration a random digit is generated and concatenated it to the end of the string.

The other buttons “Generate 1”, “Generate 10”, “Generate all digits” and “Check” are handled by the functions `generateOne()`, `generateTen()`, `generateAllDigits()`, `checkAnswer()` respectively. Currently these buttons do not perform any actions and you need to write your code according to the requirements specified by the following tasks to make the buttons work.

## Task 1.2 Generate a single clue

Save your html file under a new name called `lab12-task1-2.html`. In this task, you need to write the code in the function `generateOne()` to handle the event when the “Generate 1” button is clicked. It should generate one line showing 4 random digits and specify the result about what this quantity is equal to. The result should be displayed on the webpage in exactly the same way as shown below:



To generate 4 random digits, we can just reuse the function `generateRandom4DigitText()`. However, we also need to evaluate the result of the 4 digits to show the number on the right hand side. Probably you have already figured out from the cases on p.1 to derive a rule for generating the resulting number for each case of 4 digits. The rule is indeed quite simple, just count the number of closed regions in each digit and add them together. In the above example, the digits 5, 1 and 2 do not have any closed regions but the digit 6 has 1 closed region. So when you add up these numbers  $0+0+0+1$ , you will get 1 as the result of the right hand side. If you examine each digit from 0 to 9, then you will find that:

- The digits 1,2,3,5,7 do not have any closed regions so each of these digits has no contribution to the resulting number.
- The digits 0,4,6,9 have 1 closed region in each case so each of these digits adds 1 to the resulting number.
- The digit 8 has 2 closed regions so this digit adds 2 to the resulting number.

Now check your understanding by applying the above rules to the cases shown on p.1 and verify that you do get the same resulting number as shown on the right hand side of each case on p.1.

We will now write the Javascript code that computes the resulting number given 4 random digits. We will write it as a function `evaluateClue(clue)` that has already been declared in the start code. Note that here there is a parameter `clue` inside the function `evaluateClue`. This means that when this function is called, we can provide a parameter as an input to this function. As an analogy in Scratch, you can think of `evaluateClue` as the name of the custom block that you create and `clue` as a number input that you add when you create this custom block.

In this problem, the parameter `clue` is a string containing 4 digits. Thus it is a string with 4 characters. In the function `evaluateClue`, try to write the code by using a for-loop so that given the input parameter `clue` (a string with 4 characters containing the 4 random digits), the resulting number is computed and stored in the variable `N` to be returned at the end of the function. There can be many ways to write the code. Try to do it in your own way if you can. If not, then you can follow the flow below as one of the possible ways:

Write a for-loop that runs with 4 iterations. Initialize `N` to 0 (in addition to the counting variable in the for-loop). In each iteration,

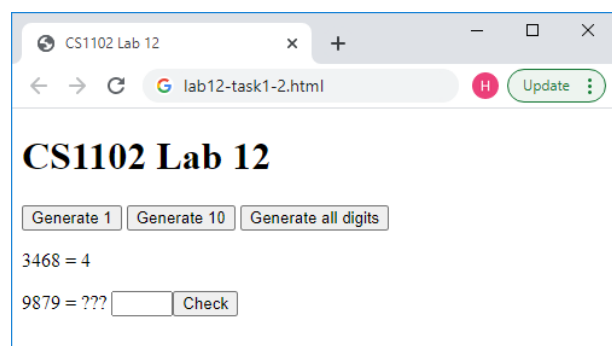
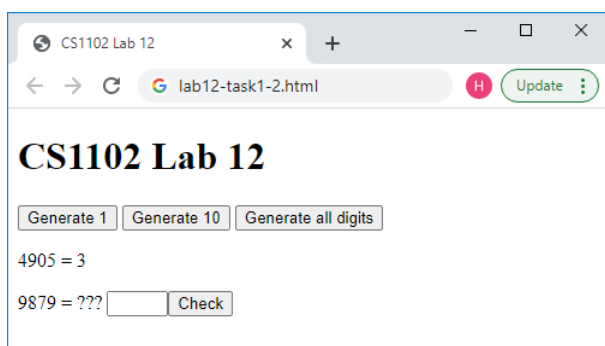
1. A character should be extracted from the string `clue`. You can consider a string as an array of characters, so you can use `clue[i]` to refer to each character with `i` from 0 to 3. For example, if `clue="5126"` as in the above example, then `clue[0]='5'`, `clue[1]='1'`, `clue[2]='2'`, `clue[3]='6'`.
2. The numeric value of the digit represented by the character can then be determined. You can use `Number()` function for this. For example, if you set `c=clue[i]`; `d=Number(c)`; then as in the same example above, when `i=0`, `c='5'` and `d=5`.
3. Add the appropriate value to `N` depending on the numeric value of a digit `d` determined from step 2. For example, if `d=1`, then you do not need to add anything to `N`; if `d=4`, then you need to increase `N` by 1; if `d=8`, then you need to increase `N` by 2. Although there are 10 possible values of `d`, you do not need to use a 9-way conditional because under many cases you do not need to change `N` (`d=1,2,3,5,7`) and in some cases you do exactly the same thing, i.e. increase `N` by 1 (`d=0,4,6,9`). Use the OR operator `||` to join these cases to form a single compound Boolean expression and write your conditional statements with just a few lines as follows:

```
if (d==0 || [expression2] || [expression3] || [expression4])
    N++;
else if ( [expression 5] )
    N=N+2;
```

Now complete this task by writing the code in the function `generateOne()`. Here you need to generate 4 random digits, evaluate the resulting number and display the result on the webpage. Try to do it in your own way if you can. If not, then you can follow the steps below:

4. Declare a variable `clue` and assign it as the result of the function call `generateRandom4DigitText()`.
5. Declare another variable `result` and assign it as the result of the function call `evaluateClue(clue)`.
6. Display the result in the right format (e.g., `5125 = 1` as in the above example) and use it as the text content for the HTML span tag with id `clue_area`, i.e.,  
`document.getElementById("clue_area").innerHTML = clue + " = " + result;`

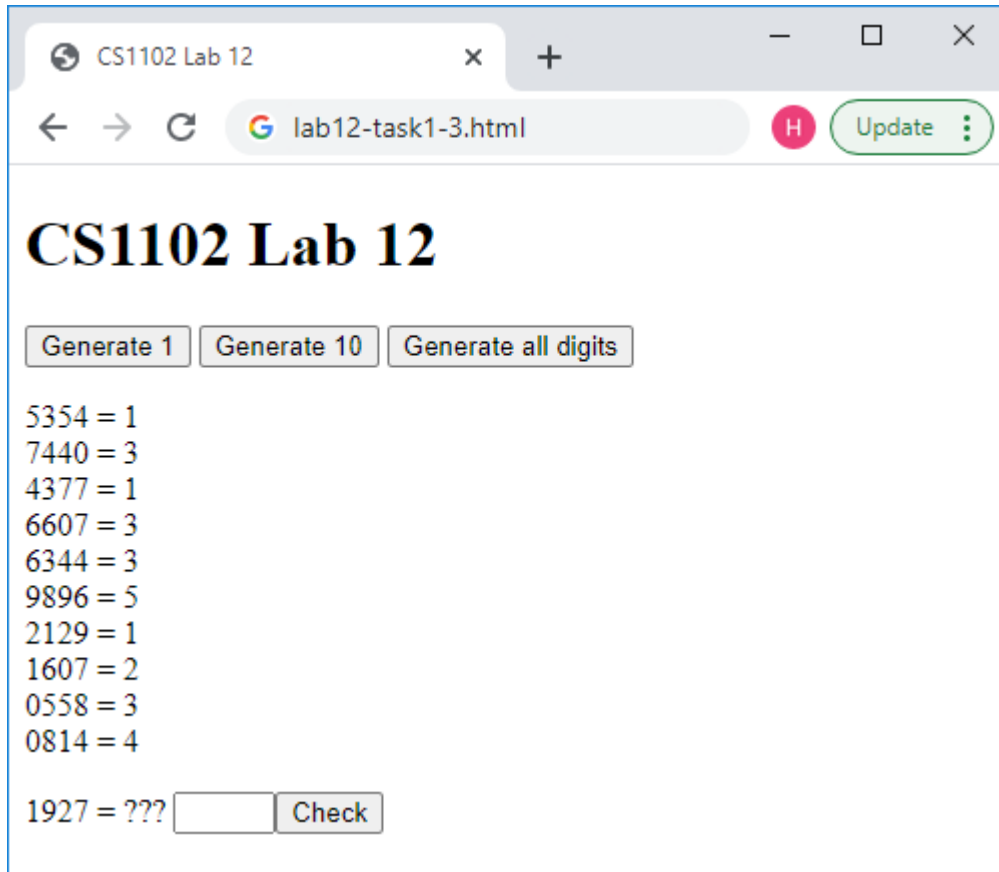
Click the button "Generate 1" a few times to verify that it is indeed giving a random clue each time and that the number shown in the right hand side is correct in each case.



Remember to save your file.

### Task 1.3 Generate 10 clues using for-loop

Save your html file under a new name called `lab12-task1-3.html`. In this task, you need to write the code in the function `generateTen()` to handle the event when the “Generate 10” button is clicked. It should use a *for-loop* to generate 10 lines of clues where each line is generated in a similar way as in Task 1.2. Your webpage content should be displayed in exactly the same way as shown below:



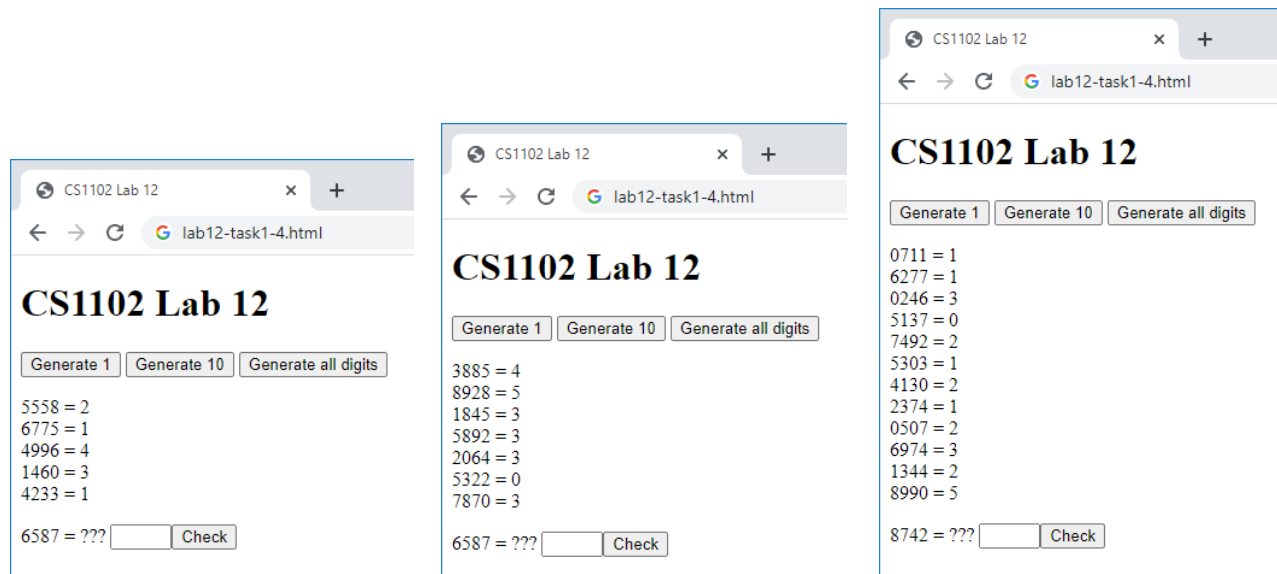
Hints (for your reference only and you may write your code in other ways):

1. Use a variable `s` to store the text content for the HTML span tag with id `clue_area`.
2. Initialize `s` as an empty string.
3. Write a for-loop with 10 iterations.
4. In each iteration, call the functions `generateRandom4DigitText` and `evaluateClue` in the same way as in Task 2 to come up with the formatted line. Then concatenate each formatted line to `s` and include `<br/>` to add the line break, e.g.,  
`s=s+[your expression]+"<br/>"`;
5. Display the text content stored in `s` for the HTML span tag with id `clue_area`, i.e.,  
`document.getElementById("clue_area").innerHTML = s`

Remember to save your file.

## Task 1.4 Generate clues until all digits are present using while-loop

Save your html file under a new name called `lab12-task1-4.html`. In this task, you need to write the code in the function `generateAllDigits()` to handle the event when the “General all digits” button is clicked. It should use a *while-loop* to generate the clues until each digit has occurred at least once. Your webpage should be displayed on the browser in exactly the same way as shown below:



Note that the number of lines may vary each time you run it but every digit from 0 to 9 shows up at least once on all the random 4 digits on the left hand side.

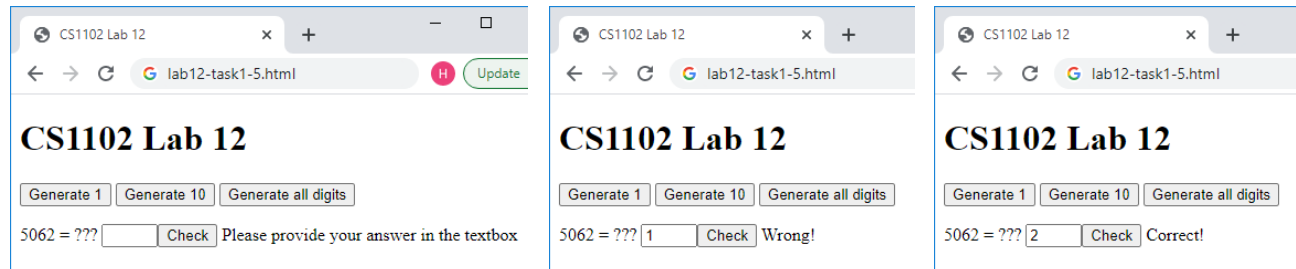
Hints (for your reference only and you may write your code in other ways):

1. As we need to determine whether every digit from 0 to 9 shows up at least once, we will use an array with 10 elements to keep track of each digit. In particular, we will use an array `countDigit` to count the number of occurrence of the corresponding digit.
2. We need to initialize every element of `countDigit` to 0. Use a for-loop with 10 iterations for this.
3. Use a Boolean variable (e.g, `isDone`) to indicate whether the termination condition is satisfied, such that it is true when all counts are at least 1 and false if at least one count is 0. Initially this variable should be set to false (indicating that it is NOT done).
4. When writing the while-loop, you need to provide the continuation condition, which should be the negation of the termination condition, thus you can use `!isDone` as the continuation condition.
5. In each iteration of the while loop,
  - a) you can first perform the same steps as in Task 3 by calling the functions `generateRandom4DigitText` and `evaluateClue`. Afterwards, you need to update the count of the newly generated digits stored in `result`. You can extract each of the 4 digits from `result` using the same approach as in Step 2 of Task 2 on p.4. Then you can increment the count of the corresponding digit by 1, e.g., `countDigit[d]++`;
  - b) after updating the count for the newly generated 4 digits, you need to set the variable `isDone` before the end of the iteration. This variable `isDone` should be set to true if all digits occur at least once (i.e., with `count > 0`) and false if a digit never occurs (i.e., with `count` equal to 0). You can use a for-loop with 10 iterations. The variable `isDone` can be initialized to true before this for-loop. In each iteration of the for-loop, check the count of the corresponding digit. If the count of a digit is 0, then set `isDone` to false.

Remember to save your file.

## Task 1.5 Check the user's answer

Save your html file under a new name called `lab12-task1-5.html`. In this task, you need to write the code in the function `checkAnswer()` to handle the event when the "Check" button is clicked. It should check whether the user has provided an input, and whether the user input is the correct answer or not, and then display the corresponding message on the webpage. Your webpage should be displayed on the browser in exactly the same way as shown below:



Hints (for your reference only and you may write your code in other ways):

1. The function `checkAnswer()` has the following line that extracts the current content of the textbox and store it to the variable `answer` as a string.  

```
var answer = document.getElementById("textboxanswer").value;
```
2. We need to check whether the answer entered by the user is the correct one. The 4 digits on the left hand side of the question are stored in the variable `clueQuestion`. You can call the function `evaluateClue(clueQuestion)` to get the correct answer. Recall that as explained in Task 1, the variable `clueQuestion` is declared as a global variable. Its value is set in the function `init()` and you can read its value in another function `checkAnswer()`. On the other hand, if you declare this variable as a local variable, i.e., if you declare it inside the function `init()`, then you will not be able to get the value of this variable in another function `checkAnswer()`.
3. The correct answer returned by `evaluateClue(clueQuestion)` is a numeric value. The user input stored in the variable `answer` is a string. So you can first convert this string to a number by using the function `Number()`. And then you can check if the user input is the correct answer with the Boolean expression `Number(answer) == evaluateClue(clueQuestion)`. If the answer is correct, then the message "Correct!" should be shown as the text content of the HTML span tag with id `feedback`. If the answer is wrong, the message "Wrong!" should be shown instead.
4. To check whether the user has entered something, the simplest way is to use the Boolean expression `answer==""` to see if the variable contains an empty string. This can work in most cases. However, for the case where the correct answer is 0 (e.g., 1235), and if the user has entered a space in the textbox, then `answer==""` is false but `Number(answer)` is equal to 0 which is the same as the correct answer. To fix this issue, the Boolean expression can be modified as `answer.trim()==""` where the function `trim()` would remove all the white spaces at the beginning and at the end of the string so it will consider the user has not entered anything if the user only enters space in the textbox. In this case, the message "Please provide your answer in the textbox" should be shown as the text content of the HTML span tag with id `feedback`.
5. You should use conditional statements to show the 3 possible messages "Correct!", "Wrong!" or "Please provide your answer in the textbox", depending on the corresponding condition. You should think about sequence in checking for the conditions to ensure a correct logic flow.

Remember to save your file.

## **Task 2    Review Previous Labs**

You can review the previous labs by reading and following the lab instructions, and check that you are able to complete the lab tasks and exercises by yourself. If you have any questions, you can ask the lab tutor or TA during the lab session.

---

## **Task 3        Challenge your classmates**

You can first reflect on what you have learnt in this lab, and then come up with problems to challenge your classmates. You can post your problem on the Canvas course page, under this [Discussion page](#). One should be able to solve your problem by using what he/she learns in Lab 12. You will not get extra marks by posting a challenging problem or solving a challenging problem posted by another student, but you will earn your fame so that you can impress the course leader, the lab tutors, and your classmates.

---

**There is NO assessment for this lab.**

---