Lab 11 Javascript 4 - Array

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.

Task 1.1 Create a New HTML File from Template in Komodo Edit

Create a new HTML file from Komodo Edit in the same way as described in Lab 08 Task 1.1:

1. Open Komodo Edit and select “New File from Template” (or press the shortcut keys Ctrl+Shift+N)
2. A dialog box will pop up on the right. Select “HTML 5” from the list

Remember to save the file after you make changes before opening it on a browser to view it.

Note that you should NOT copy and paste any code from this document to your code editor. Type up the code instead, otherwise it may not work properly, especially with symbols such as quotation marks “ ”.

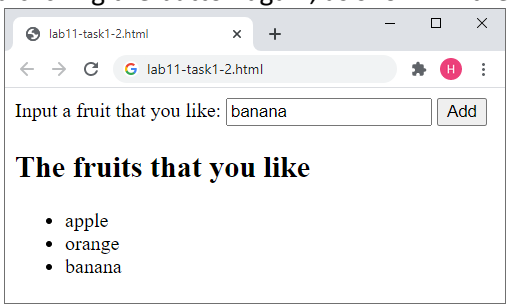
Task 1.2 List the Fruits

Download the file lab10-task1-4.html from the Canvas course page. You will see the following code if you open the file on Komodo Edit:

```
<!DOCTYPE html>

<html>
<head>
    <title></title>
    <script>
            var fruit=[];
            function addFruit() {
               var s, i;
               fruit[fruit.length] = document.getElementById("textinput").value;
               s = "<ul>";
               for (i=0; i<fruit.length; i++) {
                    s += "<li>"+fruit[i]+"</li>";
               }
               s += "</ul>";
               document.getElementById("result").innerHTML = s;
            }
    </script>
</head>
<body>
    Input a fruit that you like:
    <input type="text" id="textinput" />
    <button id="button_add" onclick="addFruit();">Add</button>
    <h2>The fruits that you like</h2>
    <div id="result"></div>
</body>
</html>
```

When you open this html file on a browser, you will see a textbox and a button. You can enter a fruit and click the button, then you will see this fruit appears as an item in the unordered list. You can keep entering another fruit and clicking the button again, as shown in the following screenshot:



Here are some explanations about the code:

Line 9: `var fruit = [];`

`fruit` is initialized as an empty array. Note that this statement is not inside the function `addFruit()` so that this line will only be called once at the beginning.

Line 10: `fruit[fruit.length] = document.getElementById("textinput").value;`

The right hand side `document.getElementById("textinput").value` is referring to the text on the textbox corresponding to the HTML element on line 22: `<input type="text" id="textinput" />`.

`fruit.length` is the size of the array `fruit`. At the beginning, the array `fruit` is initialized as the empty array by line 9. In this case `fruit.length` would be equal to 0. After this line is executed for the first time, `fruit[0]` would be assigned with the user input on the textbox, and so now the array `fruit` would have one element. The next time this line is executed, then `fruit.length` would be equal to 1.

Note that the number in the square bracket `[]`is the index of an array element, which starts at 0 as the index for the first array element.

Line 11-15:

These lines create an unordered list. The for-loop adds each element in the array `fruit` as a list item.

Line 22: `<input type="text" id="textinput" />`

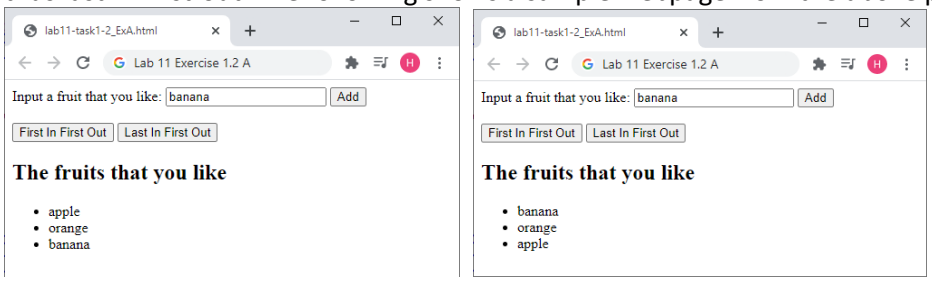
This creates a textbox on the webpage.

Line 23: `<button id="button_add" onclick="addFruit();">Add</button>`

This creates a button with the label "Add" on it. When this button is clicked, the Javascript code defined by the event handler `onclick` will be executed, i.e., the function `addFruit()` will be called.

Exercise 1.2: Modify the program so that it satisfies the following specifications. You should focus on one specification at a time, test the program each time after you finish with one specification to make sure that it is working correctly before moving to work on the next specification:

a) Add two buttons labelled as "First In First Out" and "Last In First out". When the first button is clicked, the order of the fruits displayed is the same as stored in the array, thus first-in-first-out. When the second button is clicked, the order of the fruits displayed is the reverse of the order stored in the array, thus last-in-first-out. The following shows a sample webpage from the above program:



Hint 1: the onclick event handler for the first button can call a function with the code which is very similar to the function `addFruit()`.

Hint 2: the onclick event handler for the second button can call a function with the code which is very similar to the function defined as the event handler for the first button, with the difference in the for-loop statement, i.e., the initialization, continuation condition, increment statement. How can you change just this statement so that the value of `i` is iterated from the highest to the lowest?

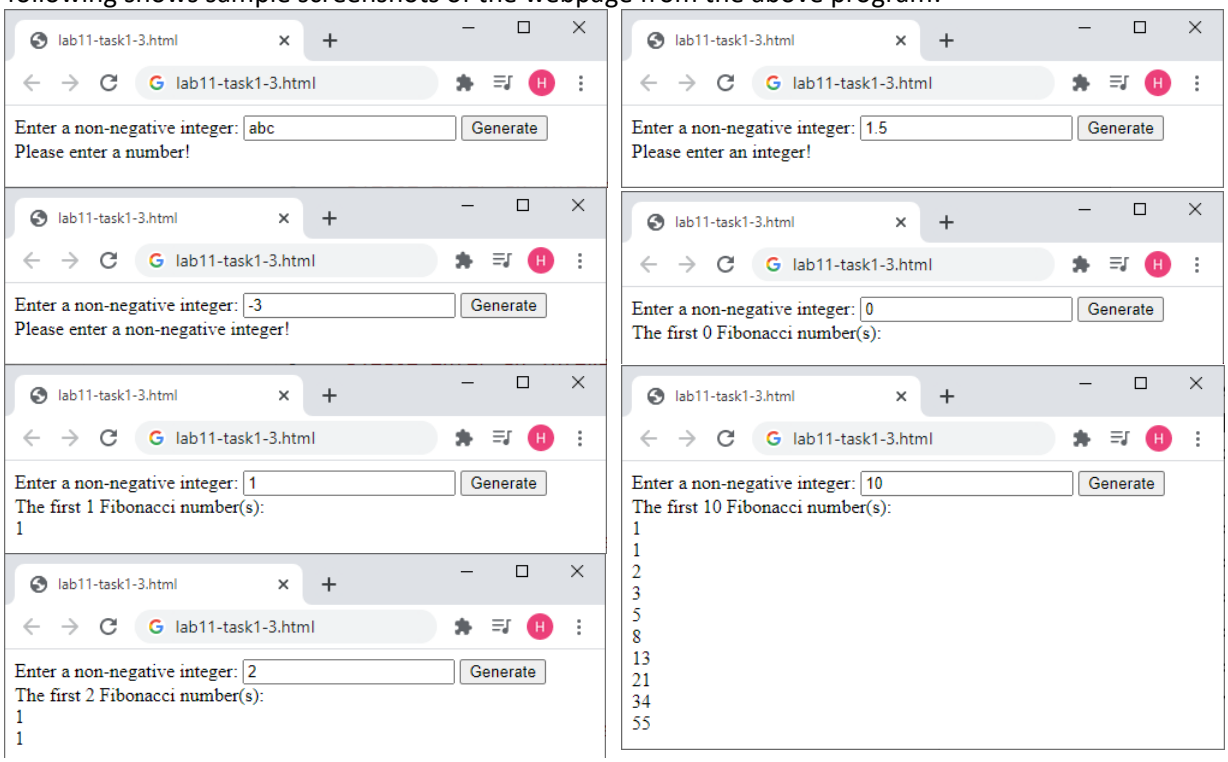
b) Modify your program from a) such that instead of showing all the fruits when the "First In First Out" or "Last In First out" buttons are clicked, only up to 4 fruits will be shown. If the user has entered less than 4 fruits, then all of them will be displayed in the same way as in a). On the other hand, if the user has entered more than 4 fruits, and when the "First In Last Out" button is clicked, then only the first 4 elements in the array will be displayed. If the user has entered more than 4 fruits, and when the "Last In Last Out" button is clicked, then only the last 4 elements in the array will be displayed.

Task 1.3 Calculate Fibonacci Numbers

In this task you will write a program to ask the user to input a non-negative integer N and validate if the user input is indeed an integer. Once the input has been validated, then the first N Fibonacci numbers will be displayed. The Fibonacci numbers are defined such that the first 2 numbers are both 1 and then the subsequent Fibonacci number is obtained by summing up the previous 2 Fibonacci numbers. So if we denote F(n) as the n-th Fibonacci number, then

F(1) = 1
F(2) = 1
F(3) = F(1) + F(2) = 1 + 1 = 2
F(4) = F(2) + F(3) = 1 + 2 = 3
F(5) = F(3) + F(4) = 2 + 3 = 5
So the sequence of Fibonacci numbers are 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ……

The following shows sample screenshots of the webpage from the above program:



Now try to come up with the program by yourself.

We now show you a sample HTML and Javascript codes implemented according to the above requirements for your reference:

```
1  <!DOCTYPE html>
2
3  <html>
4  <head>
5      <title></title>
6      <script>
7              function generateFibonacci() {
8                 var F=[1,1];
9                 var input, s;
10                input = document.getElementById("textinput").value;
11                if (isNaN(input)) {
12                   s = "Please enter a number!";
13                }
14                else {
15                   N = Number(input);
16                   if (N != Math.floor(N)) {
17                       s = "Please enter an integer!";
18                   }
19                   else if (N < 0) {
20                       s = "Please enter a non-negative integer!";
21                   }
22                   else {
23                      for (i=2; i<N; i++) {
24                          F[i] = F[i-1]+F[i-2];
25                      }
26                      s = "The first "+N+" Fibonacci number(s):<br/>";
27                      for (i=0; i<N; i++) {
28                          s += F[i]+"<br/>";
29                      }
30                   }
31                }
32                document.getElementById("result").innerHTML = s;
33             }
34     </script>
35 </head>
36 <body>
37     Enter a non-negative integer:
38     <input type="text" id="textinput" />
39     <button id="button_generate" onclick="generateFibonacci();">Generate</button>
40     <div id="result"></div>
41 </body>
42 </html>
```

Here are some explanations about the code:

Line 8: `var F=[1,1];`

`F` is initialized as an array with 2 elements with the value of 1 for each element.

Line 11-21:

These conditionals validate the input and assign the corresponding message to the variable `s` if the input is not a non-negative integer.

Line 23-25:

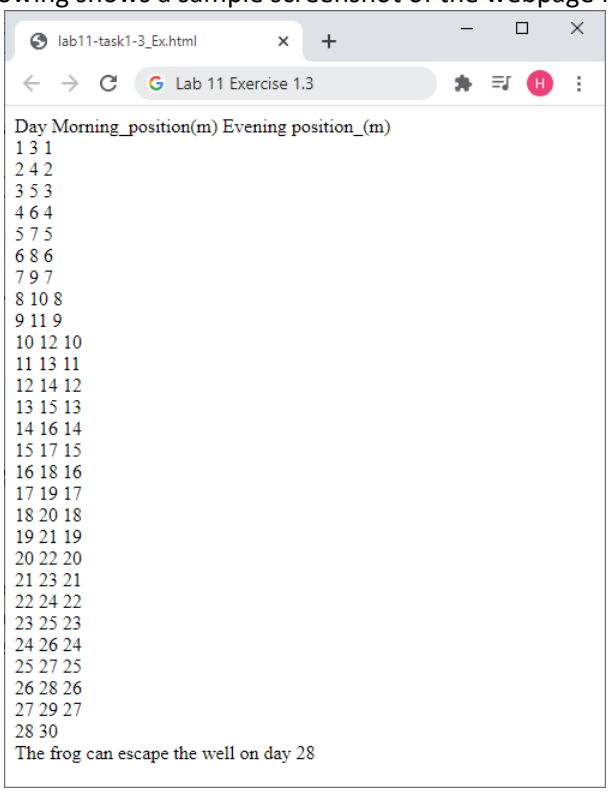
This for-loop calculates the Fibonacci numbers. Note that the index of `i` starts with 2 (instead of 0) as the first 2 Fibonacci numbers have already been defined in line 8. You should think about how many iterations will this for-loop run if N=0,1,2,10 respectively.

Line 26-29:

The resulting Fibonacci numbers are concatenated with the line break tag `<br/>` to be assigned to the variable `s`.

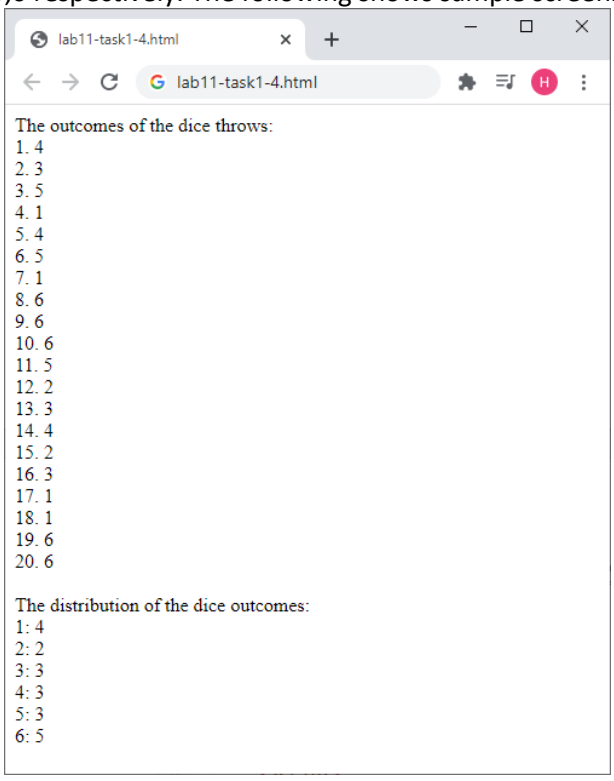
Exercise 1.3: Write a program to solve the following problem: a frog has fallen at the bottom of a well that is 30m deep. Every morning, the frog can climb up 3m. However, in the evening, the frog will fall down 2m. How many days will it take for the frog to reach the top of the well and escape? On the webpage, display the day number, the position of the frog in the morning, the position of the frog in the evening on each line until the frog has reached the top and is able to escape. Hint: use 2 arrays to store the positions of the frog in the morning and in the evening and use a loop to keep adding new elements to these arrays until the frog has reached the top of the well. Then display the results from these arrays on the webpage.

The following shows a sample screenshot of the webpage from the above program:



Task 1.4 Compute Distributions of Dice Throws

In this task you will write a program to generate 20 random dice throws. The dice throw outcomes will be shown first, followed by the distribution of the dice outcomes, i.e., the number of times the dice outcome is 1,2,34,5,6 respectively. The following shows sample screenshots of the webpage from the above program:



Now try to come up with the program by yourself.

We now show you a sample HTML and Javascript codes implemented according to the above requirements for your reference:

```
1  <!DOCTYPE html>
2
3  <html>
4  <head>
5      <title></title>
6      <script>
7              function init() {
8                 var sum=[];
9                 var i, d;
10                for (i=1; i<=6; i++) {
11                  sum[i] = 0;
12                }
13                s = "The outcomes of the dice throws:<br/>";
14                for (i=1; i<=20; i++) {
15                  d = Math.floor(Math.random()*6)+1;
16                  sum[d]++;
17                  s += i+". "+d+"<br/>";
18                }
19                s += "<br/>The distribution of the dice outcomes:<br/>";
20                for (i=1; i<=6; i++) {
21                  s+=i+": "+sum[i]+"<br/>";
22                }
23                document.getElementById("result").innerHTML = s;
24             }
25     </script>
26 </head>
27 <body onload="init();">
28     <div id="result"></div>
29 </body>
30 </html>
```

In the above program, the array `sum` is used to store the distribution of the dice outcomes 1-6. The elements of `sum` need to be initialized to 0 in line 10-12. Line 16 increases the count of an element in `sum` by 1 which corresponds to the dice outcome at that iteration.

Exercise 1.4: Modify the program so that it satisfies the following specifications. You should focus on one specification at a time, test the program each time after you finish with one specification to make sure that it is working correctly before moving to work on the next specification:

a) Instead of throwing one dice, two dice will be thrown each time. The outcomes of the 2 dice as well as their sum will be displayed. The distribution of the sum of the dice will be shown at the end. Note that with 2 dice, the sum ranges from 2 to 12.

b) Modify the program from a) such that instead of throwing the two dice for 20 times, keep throwing them until each sum has appeared at least once.

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 Complete the assessment from the Canvas course page

You should complete the Lab 11 Assessment from the Canvas course page before the posted deadline.

Task 4 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 11. 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.