CS1102
Lecture 5 (Part B)
Web Development: Javascript

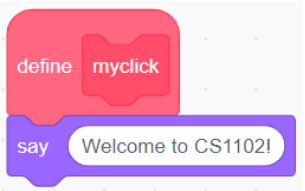
Semester B, 2021-2022
Department of Computer Science
City University of Hong Kong

Javascript

- Javascript is a programming language that can provide instructions for a browser to dynamically generate content for a website or enhance the website interactivity
- Javascript can be embedded in the head section of the webpage, with the code defined inside the `<script> </script>` tags

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>JS 1st Example</title>
5      <script>
6          function myclick() {
7              alert("Welcome to CS1102!");
8          }
9      </script>
10   </head>
11   <body>
12   <!-- Page content begins here -->
13     <h1>CS1102</h1>
14     <h2 onclick="myclick();">Click Me</h2>
15     <!-- Page content ends here -->
16   </body>
17 </html>
```

In Scratch, you drag and drop the blocks to form your code.
In Javascript, you have to type the code and you need to be careful to avoid typos which will cause errors in your program.



The semi-colon `;` is placed at the end of each Javascript statement

Javascript: Website Interactivity

- Website interactivity can be enhanced by detecting a user event and defining the corresponding event handler to perform certain action, e.g., the following program has an event handler that detects if the user clicks on the h2 heading "Click Me" and then calls the function `myclick` to pop up a message.

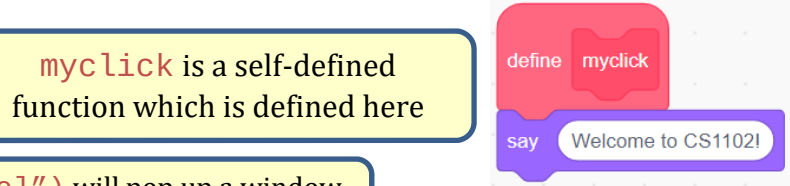
```
1  <!DOCTYPE html>
2  <html>
3    <head>
8      <title>Javascript First Example</title>
9      <script>
10       function myclick() {
11         alert("Welcome to CS1102!");
12       }
13     </script>
14   </head>
15   <body>
16   <!-- Page content begins here -->
17     <h1>CS1102</h1>
18     <h2 onclick="myclick();">Click Me</h2>
19     <!-- Page content ends here -->
20   </body>
21 </html>
```

`myclick` is a self-defined function which is defined here

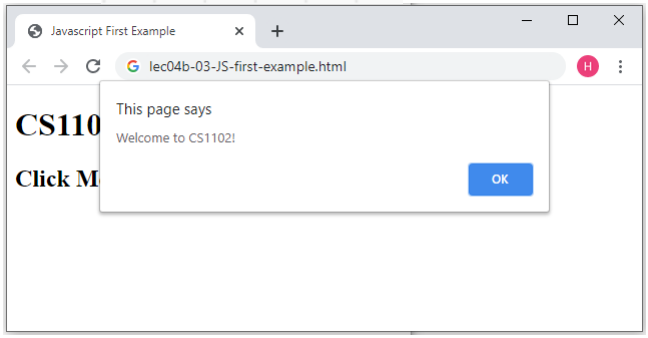
The statement `alert("[message]")` will pop up a window and display the message specified inside the parentheses `()`

The attribute `onclick` is an event handler that is invoked when this h2 element is clicked. In this case, the Javascript function `myclick()` will be called

Scratch Analogy



Scratch Analogy



Code Example:
lec04b-03-JS-first-example.html

Javascript: Dynamic Content

- The innerHTML property allows the content of an HTML element to be changed dynamically

1. Originally the "welcome" div has no content: `<div id="welcome"></div>`
2. After the page has finished loading, the `onload` function is invoked which will call the `showDynamicContent()` function
3. The following statement
   `document.getElementById("welcome").innerHTML = "<h2>Welcome!</h2>"`
   replaces the current content of the "welcome" div with the string `"<h2>Welcome!</h2>"` such that the webpage will be displayed as if the html of the "welcome" div is
   `<div id="welcome"><h2>Welcome!</h2></div>`

```
1  <!DOCTYPE html>
2  <html>
3    <head>
8      <title>Javascript Dynamic Content</title>
9      <script>
10       function showDynamicContent() {
11         document.getElementById("welcome").innerHTML="<h2>Welcome!</h2>";
12       }
13     </script>
14   </head>
15   <body onload="showDynamicContent();">
16   <!-- Page content begins here -->
17     <h1>CS1102</h1>
18     <div id="welcome"></div>
19     <!-- Page content ends here -->
20   </body>
21 </html>
```

Code Example: lec04b-04-JS-innerHTML.html

Usually the `onload` event handler is added as an attribute in the body tag and is used to call some Javascript function to carry out some tasks right after the webpage has finished loading to initialize some settings

Comments

- HTML, CSS and Javascript allow programmers to insert comments in the code but their syntax differs
- Comments will be ignored by the browser when the webpage is displayed but it is a good practice to insert comments to document what the code logic, such that it will be easy for others to understand your code or for you to revisit your code later

In CSS, comments can be placed between /* and */ and can span multiple lines

In Javascript, there are 2 ways to add comments:
1) Similar to CSS, Javascript comments can be placed between /* and */ and can span multiple lines
2) Javascript comments can also be placed after // until the end of line so this is a single line comment. Note that CSS does not support this single line comment style

```
<!DOCTYPE html>
<html>
  <head>
    <title>Comments</title>
    <style>
        /* The following CSS style sets the corresponding div element
           with color red */
        #course {
            color: red;
        }
    </style>
    <script>
        function init() {   // this function is called after the webpage has been loaded
            /* The following statment dynamically replaces the content
              of the corresponding div elemnt by the given string */
            document.getElementById("course").innerHTML="<h2>Introduction to Computer Studies</h2>";
        }
    </script>
  </head>
  <body onload="init();">
  <!-- Page content begins here -->
    <h1>CS1102</h1>
    <!-- the following div element's content is empty in the HTML
         and will be assigned by Javascript after the webpage has been loaded -->
    <div id="course"></div>
  <!-- Page content ends here -->
  </body>
</html>
```

In HTML, comments can be placed between <!-- and --> and can span multiple lines

Code Example: lec04b-05-comment.html

Javascript: Prompting User Input

- Similar to the block 

in Scratch, Javascript can use the `prompt()` function to actively ask the user to input something

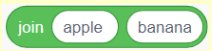
In Javascript, variables can be declared by using the syntax `var` followed by the variable name. There are several rules for defining valid variable names:

1. Names are case sensitive, e.g., `name` and `Name` are different variables
2. Names can begin with letters, underscore `_` , dollar sign `$`, but not digits
3. Digits can be used in names as long as it is not the first character
4. Names cannot be the same as reserved words such as `var`, `prompt`, etc.

Javascript Reserved Words:
https://www.w3schools.com/js/js_reserved.asp

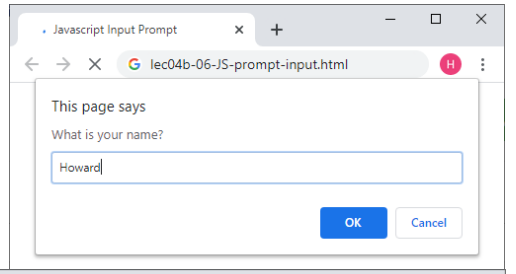
```
<!DOCTYPE html>
<html>
  <head>
    <title>Javascript Input Prompt</title>
    <script>
        function init() {
            var name;  // declare a variable
            name = prompt("What is your name?");  // assign the user input to the variable
            document.getElementById("hello").innerHTML = "Hello " + name + "!";
        }
    </script>
  </head>
  <body onload="init();">
  <!-- Page content begins here --
    <div id="hello"></div>
    <p>Welcome to CS1102!</p>
    <!-- Page content ends here --
  </body>
</html>
```

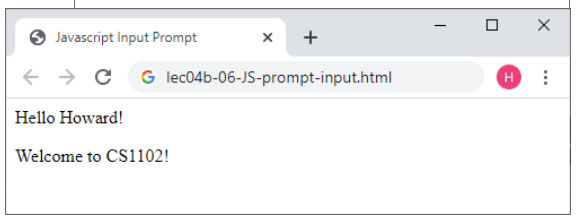
Code Example: lec04b-06-JS-prompt-input.html

"Hello" + name + "!" concatenate these strings together to from a longer string. This is similar to the block 

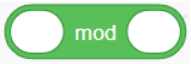
in Scratch

The prompt() function is a Javascript built-in function and it pops up a dialog box asking the user to provide input on a textbox. The output of this function is the string input by the user on the textbox



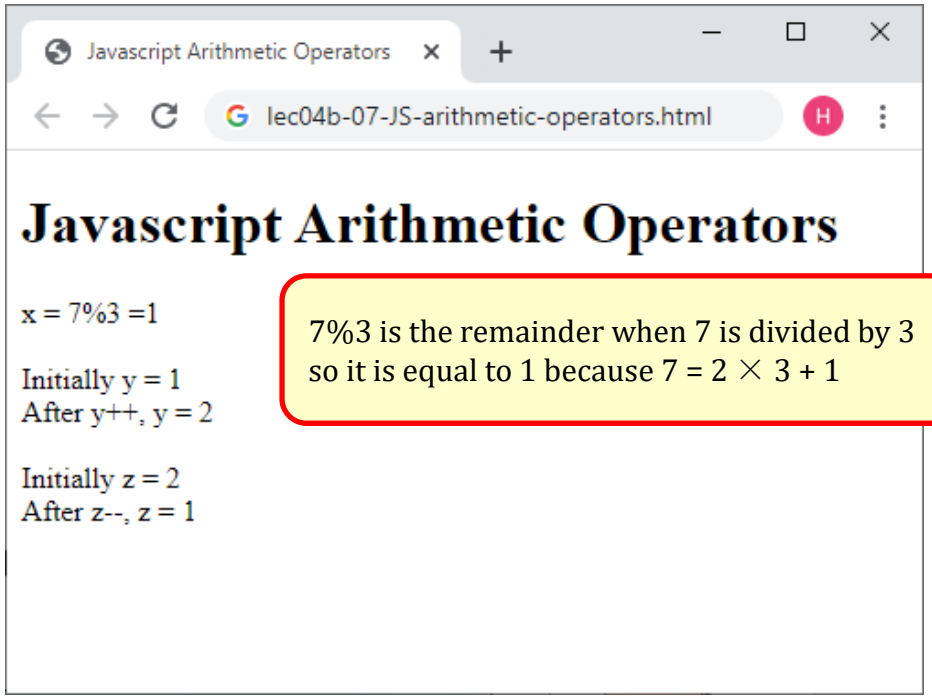


Javascript: Arithmetic Operators

- Javascript arithmetic operators:
  - The 4 operators +, -, *, / are intuitive
  - % is the modulus operator like 

  - ++ is the increment operator, x++ will increase the value of the variable x by 1
  - -- is the decrement operator, x-- will decrease the value of the variable x by 1

```
<!DOCTYPE html>
<html>
  <head>
    <title>Javascript Arithmetic Operators</title>
    <script>
        function init() {
            var s = "";
            var x,y,z;

            x = 7%3;
            s = s + "x = 7%3 =" + x + "<br /><br />";

            y = 1;
            s = s + "Initially y = " + y + "<br />";
            y++;
            s = s + "After y++, y = " + y + "<br /><br />";

            z = 2;
            s = s + "Initially z = " + z + "<br />";
            z--;
            s = s + "After z--, z = " + z + "<br /><br />";

            document.getElementById("display").innerHTML = s;
        }
    </script>
  </head>
  <body onload="init();">
  <!-- Page content begins here -->
    <h1>Javascript Arithmetic Operators</h1>
    <div id="display"></div>
  <!-- Page content ends here -->
  </body>
</html>
```



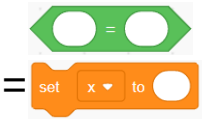
7%3 is the remainder when 7 is divided by 3 so it is equal to 1 because 7 = 2 × 3 + 1

Code Example: lec04b-07-JS-arithmetic-operators.html

Javascript: Comparison and Logical Operators

- Javascript comparison operators: <, <=, >, >=, ==, !=
  - The first 4 operators <, <=, >, >= are intuitive
  - The operator == checks whether the expressions on the left hand side and right hand side are equal. This should not be confused with the assignment operator =
  - The operator != checks whether the expressions are different (not equal to)

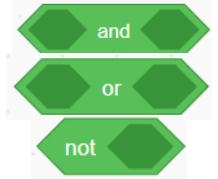
Scratch Analogy




```
<!DOCTYPE html>
<html>
  <head>
    <title>Javascript Comparison and Logical Operators</title>
    <script>
        function init() {
            var s = "";
            var input = prompt("Enter a month (1-12):");
            var month = Number(input);

            if (month>8 && month <=12)
                s = s+"This month is in Semester A\n";

            if (month == 12 || month < 3)
                s = s+"This is a winter month.\n";

            if (month != 12)
                s = s+"Santa Claus is not coming this month.\n";

            alert(s);
        }
    </script>
  </head>
  <body onload="init();">
  <!-- Page content begins here -->
    <h1>Logical Operators</h1>
    <!-- Page content ends here -->
  </body>
</html>
```

- Javascript logical operators:
  - && is the AND operator
  - || is the OR operator
  - ! is the NOT operator

Scratch Analogy



The expression (month>8 && month <=12) is true if month > 8 AND month <=12, i.e., when month is equal to 9, 10, 11, 12

The expression (month==12 || month < 3) is true if month is equal to 12 OR month < 3, i.e., when month is equal to 12, 1, 2

The expression (month!=12) is true if month is NOT equal to 12, i.e., when month is equal to 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

Code Example: lec04b-08-JS-comparison-logical-operators.html

Javascript: One-Way Conditional

Code Example: lec04b-09-JS-one-way-conditional.html

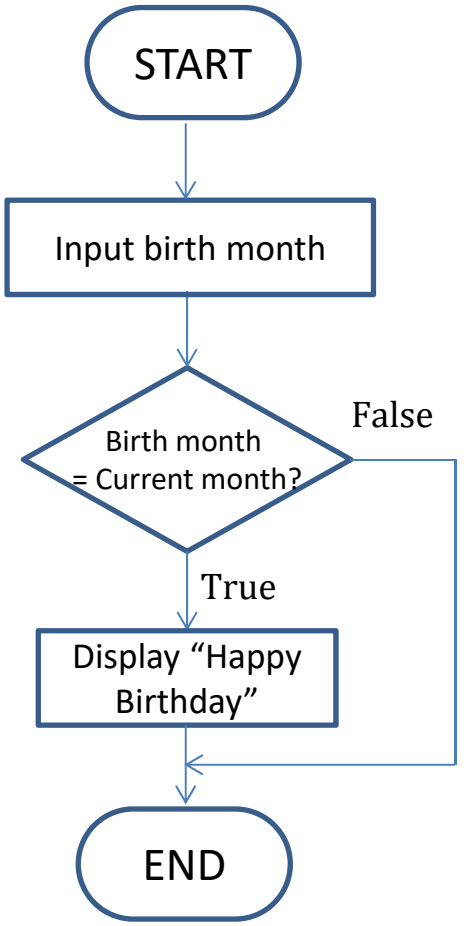
```
1  <!DOCTYPE html>
2  <html>
3    <head>
8      <title>Javascript One-Way Conditional</title>
9      <script>
10       function init() {
11         var p, s;
12         var currentDate = new Date();
13         s = "";
14         p = prompt("What is your birth month? (1-12)");
15         if (Number(p)==currentDate.getMonth()+1)
16           alert("Happy Birthday!");
17       }
18     </script>
19   </head>
20   <body onload="init();">
21   <!-- Page content begins here -->
22     <h1>One Way Conditional</h1>
23     <!-- Page content ends here -->
24   </body>
25 </html>
```

The variable `s` is initialized to be the empty string `""`

The if statement behaves in a similar way as the Scratch block

The expression `currentDate.getMonth()` returns the current month in the range from 0 to 11 denoting January to December.
After adding 1, the expression `currentDate.getMonth()+1` will thus have the range from 1 to 12.
This is analogous to the Scratch block

The variable `p` is assigned as a string (text) from the output of the prompt function. The expression `Number(p)` converts the string `p` to its numerical value so that it can be manipulated as a number

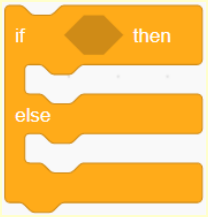


Javascript: Two-Way Conditional

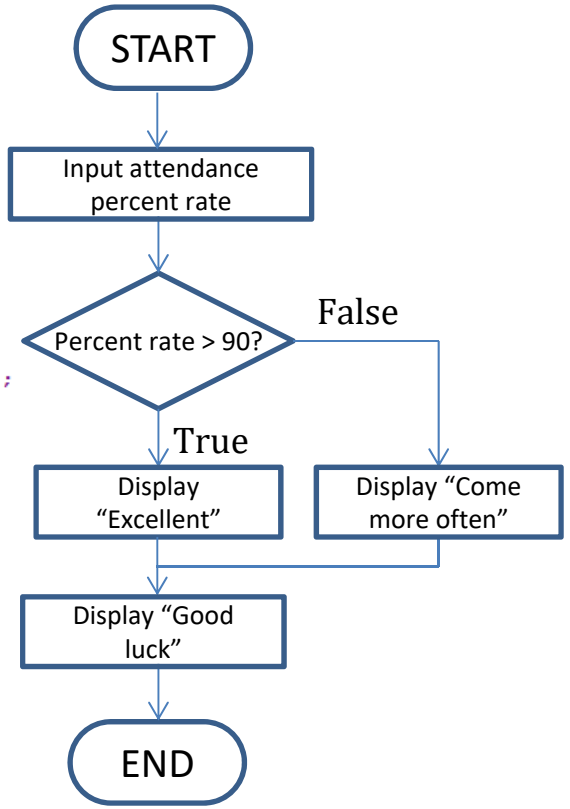
Code Example: lec04b-10-JS-two-way-conditional.html

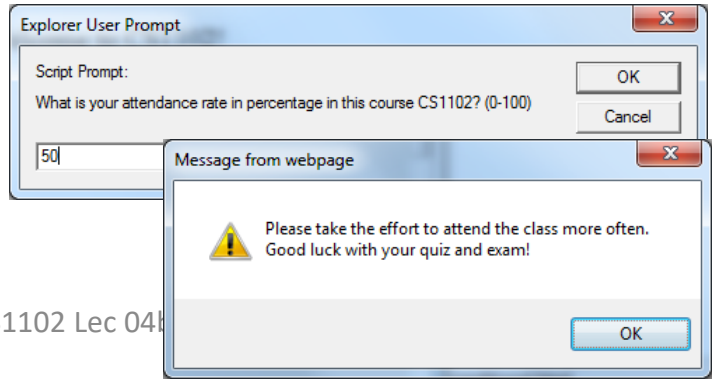
```
<!DOCTYPE html>
<html>
  <head>
    <title>Javascript Two-Way Conditional</title>
    <script>
        function init() {
            var p, s;
            s = "";
            p = prompt("What is your attendance rate in percentage in this course CS1102? (0-100)");
            if (Number(p)>90)
                s = s+"You have excellent attitude!\n";
            else
                s = s+"Please take the effort to attend the class more often.\n";
            s = s+"Good luck with your quiz and exam!";
            alert(s);
        }
    </script>
  </head>
  <body onload="init();">
  <!-- Page content begins here -->
    <h1>Two-Way Conditional</h1>
    <!-- Page content ends here -->
  </body>
</html>
```

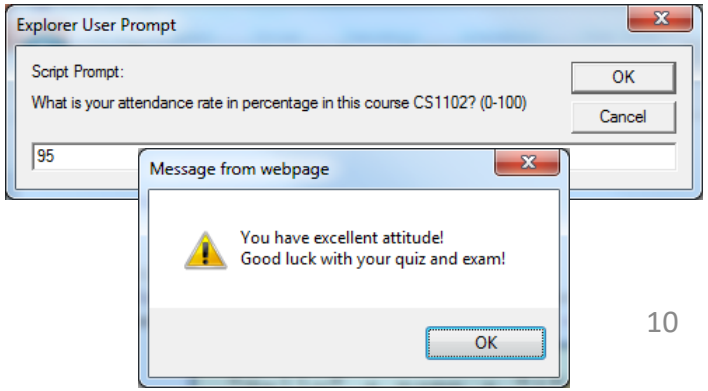
The if-else statement behaves in a similar way as the Scratch block



\n specifies that a line break should be added such that the subsequent text will be displayed in a new line when shown by the alert() function







Javascript: N-Way Conditional

Code Example: lec04b-11-JS-N-way-conditional.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Javascript N-Way Conditional</title>
    <script>
        function init() {
            var p, s, age;
            s = "";
            p = prompt("How old are you? Round to the nearest integer");
            age = Number(p);
            if ( (age >= 0) && (age <= 1) )
              s = "Infant";
            else if (age <= 4)
              s = "Toddler";
            else if (age <= 12)
              s = "Child";
            else if (age <= 19)
              s = "Teenager";
            else if (age <= 39)
              s = "Adult";
            else if (age <= 59)
              s = "Middle Age Adult";
            else
              s = "Senior Adult";
            alert(s);
        }
    </script>
  </head>
  <body onload="init();">
    <!-- Page content begins here -->
    <h1>N-Way Conditional</h1>
    <!-- Page content ends here -->
  </body>
</html>
```

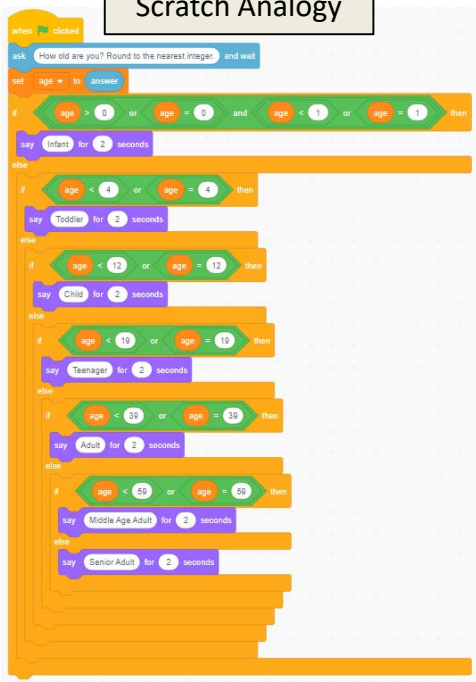
Age	Boolean Expression	Stage of Life
0-1	age<=1	Infant
2-4	age>=2 AND age<=4	Toddler
5-12	age>=5 AND age<=12	Child
13-19	age>=13 AND age<=19	Teenager
20-39	age>=20 AND age<=39	Adult
40-59	age>=40 AND age<=59	Middle Age Adult
60+	age>=60	Senior Adult

an N-way conditional can be constructed by placing one if-else statement inside another one in a *nested* manner

The program expects the user to enter a non-negative integer.

1. What message do you think the program will output if the user inputs a negative number? Try to trace the program first to anticipate the answer before verifying it by running the program with such kind of input.
2. Can you modify this program to output an error message when the input is a negative number?

Scratch Analogy



Javascript: For-Loop

- The for-loop is often used to carry out a task for a finite number of times

The for-statement, e.g. `for (i=0; i<N; i++)`, contains 3 parts inside the parentheses:

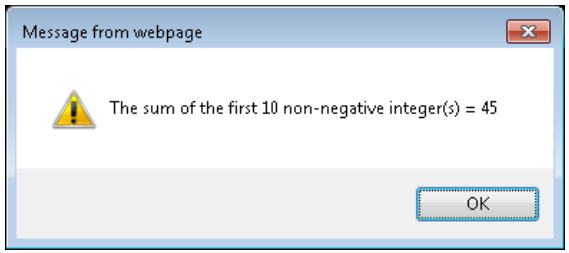
1. Initialization: the code is executed at the beginning of the loop
   e.g., `i=0` assigns `0` to the variable `i` right after the statement `sum=0;`
2. Continuation condition: the tasks specified in the loop are carried out if the continuation condition in the form of Boolean expression is true, otherwise the loop ends if the continuation condition is false
   e.g., `i<10` is true when `i=0,1,2,3,4,5,6,7,8,9` but is false when `i=10;`
3. Increment statement: this part is executed at the end of each iteration of the loop
   e.g., `i++` means that the variable `i` is increased by `1` and it is executed at the end of each iteration of the loop after the statement `sum = sum + i;`

- Note that there should be no semi-colon after the parenthesis,
  i.e. `for (i=0; i<N; i++);` is wrong

Code Example: lec04b-12-JS-for-loop.html

```
1  <!DOCTYPE html>
2  <html>
3    <head>
8      <title>Javascript For-Loop</title>
9      <script>
10         function init() {
11             var i, N, sum;
12             N = 10;
13             sum = 0;
14             for (i=0; i<N; i++) {
15                 sum = sum + i;
16             }
17             alert("The sum of the first "+N+" non-negative integer(s) = "+sum);
18         }
19     </script>
20   </head>
21   <body onload="init();">
22   <!-- Page content begins here -->
23     <p>Adding the first N integers</p>
24     <!-- Page content ends here -->
25   </body>
26 </html>
```

The curly brackets after the for-statement enclose the statements that are executed at each iteration of the for-loop, in this example, `sum = sum + i;`
You can put multiple statements inside the curly brackets such that all of them will be executed at each iteration of the loop



Javascript: For-Loop (cont.)

```
N = 10;
sum = 0;
for (i=0; i<N; i++) {
    sum = sum + i;
}
```

The code on the left is executed according to the following sequence of operations:

1. N=10
2. sum=0
3. i=0

1st iteration

4. i<N ⇔ 0<10=true so the loop will continue to run
5. sum=sum+i ⇔ sum=0+0=0
6. i++ ⇔ i=0+1=1

2nd iteration

7. i<N ⇔ 1<10=true so the loop will continue to run
8. sum=sum+i ⇔ sum=0+1=1
9. i++ ⇔ i=1+1=2

3rd iteration

10. i<N ⇔ 2<10=true so the loop will continue to run
11. sum=sum+i ⇔ sum=1+2=3
12. i++ ⇔ i=2+1=3

4th iteration

13. i<N ⇔ 3<10=true so the loop will continue to run
14. sum=sum+i ⇔ sum=3+3=6
15. i++ ⇔ i=3+1=4

5th iteration

16. i<N ⇔ 4<10=true so the loop will continue to run
17. sum=sum+i ⇔ sum=6+4=10
18. i++ ⇔ i=4+1=5

6th iteration

19. i<N ⇔ 5<10=true so the loop will continue to run
20. sum=sum+i ⇔ sum=10+5=15
21. i++ ⇔ i=5+1=6

7th iteration

22. i<N ⇔ 6<10=true so the loop will continue to run
23. sum=sum+i ⇔ sum=15+6=21
24. i++ ⇔ i=6+1=7

8th iteration

25. i<N ⇔ 7<10=true so the loop will continue to run
26. sum=sum+i ⇔ sum=21+7=28
27. i++ ⇔ i=7+1=8

9th iteration

28. i<N ⇔ 8<10=true so the loop will continue to run
29. sum=sum+i ⇔ sum=28+8=36
30. i++ ⇔ i=8+1=9

10th iteration

31. i<N ⇔ 9<10=true so the loop will continue to run
32. sum=sum+i ⇔ sum=36+9=45
33. i++ ⇔ i=9+1=10
34. i<N ⇔ 10<10=false so the loop will end

Javascript: While-Loop

- The while-loop is used to carry out a task repeatedly as long as a continuation condition is true

Code Example: lec04b-14-JS-while-loop.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Javascript While-Loop</title>
    <script>
        function init() {
            var isInputValid, number;
            isInputValid = false;
            while (!isInputValid) {
                number = prompt("Input a positive integer");
                if (isNaN(number)) {
                    alert("Please enter a NUMBER!");
                }
                else if (Number(number) <= 0) {
                    alert("Please enter a POSITIVE number!");
                }
                else
                    isInputValid = true;
            }
            alert("The positive number that you entered is "+number);
        }
    </script>
  </head>
  <body onload="init();">
  <!-- Page content begins here -->
    <p>Checking for Positive Number</p>
    <!-- Page content ends here -->
  </body>
</html>
```

`isInputValid` is a Boolean variable which has value `true` or `false`
- it is set to be `false` initially
- it will be set to `true` if the user inputs a positive number

`!` is the NOT operator and will negate its subsequent Boolean expression

isInputValid	!isInputValid
true	false
false	true

`isNaN` is Javascript built-in function and is used to check whether the given parameter is not a number. It will return `true` if it is not a number and `false` if it is a number, e.g.,

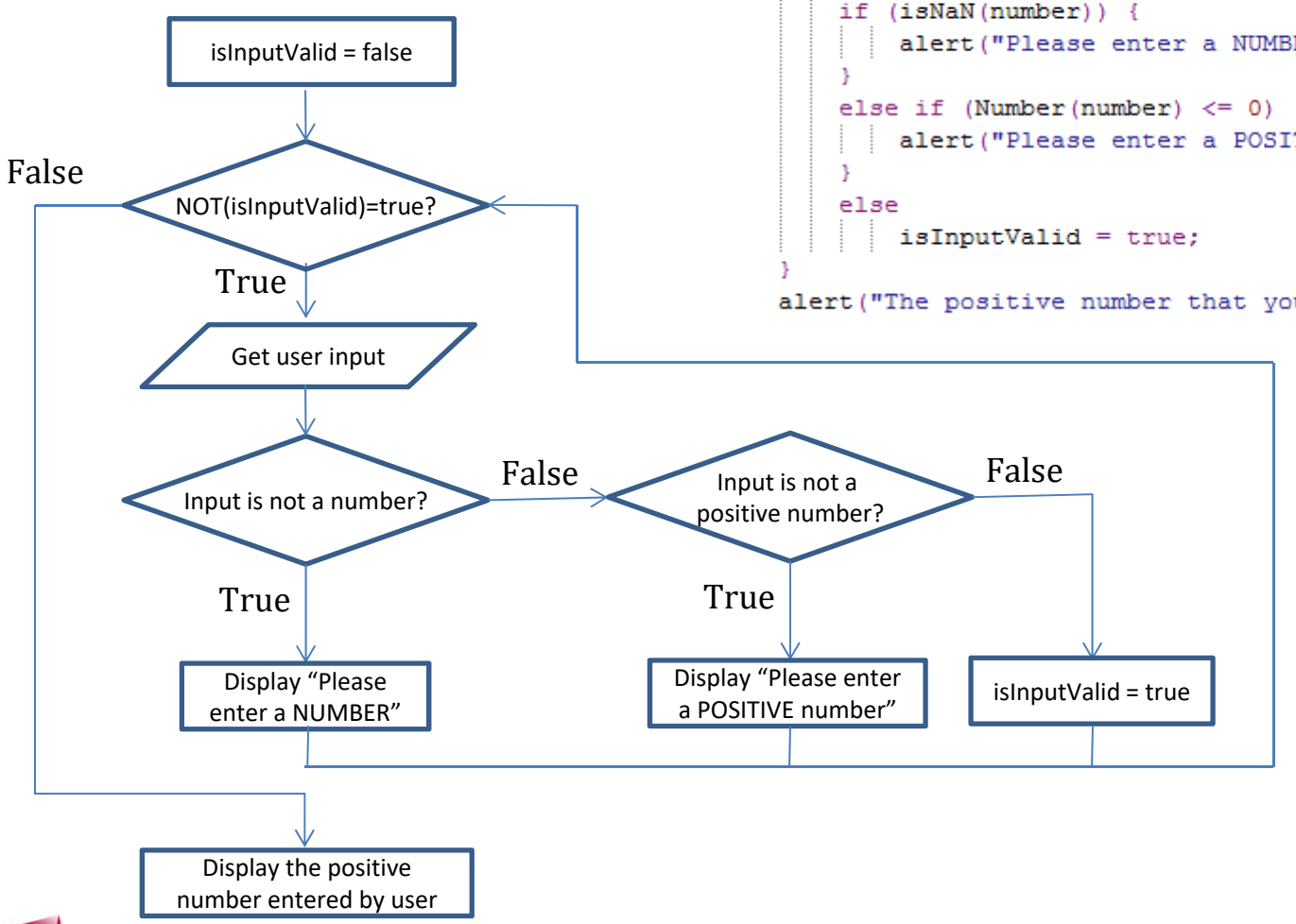
`isNaN(234)` → `false`
`isNaN("abc")` → `true`

`Number` is Javascript built-in function and is used to convert the given parameter to a number according to its value such that numeric calculations can be applied, e.g.,

`Number("123")` → `123`
`Number("123")+1` → `124`
however, `"123"+1` → `"1231"`

The curly brackets after the while-statement enclose the statements that are executed at each iteration of the while-loop

Javascript: While-Loop (cont.)




```
isInputValid = false;
while (!isInputValid) {
    number = prompt("Input a positive integer");
    if (isNaN(number)) {
        alert("Please enter a NUMBER!");
    }
    else if (Number(number) <= 0) {
        alert("Please enter a POSITIVE number!");
    }
    else
        isInputValid = true;
}
alert("The positive number that you entered is "+number);
```

Javascript: Button

- A button can be created by using the HTML tag <button>
- The attribute `onclick` can be used to handle the event when the button is clicked

Code Example: lec04b-16-JS-button.html

```
1   <!DOCTYPE html>
2   <html>
3     <head>
8       <title>Javascript Button</t
9       <script>
10        function init() {
11          document.getElementById("number").style = "font-size:100px;";
12        }
13        function throw_dice(){
14          var n;
15          n = Math.floor(Math.random()*6)+1;
16          document.getElementById("number").innerHTML = n;
17        }
18      </script>
19    </head>
20    <body onload="init();">
21    <!-- Page content begins here
22      <h1>Javascript Button</h1>
23      <button id="dice" onclick="throw_dice();" >Throw Dice</button>
24      <br/>
25      <div id="number" ></div>
26    <!-- Page content ends here -->
27    </body>
28  </html>
```

The style can be set by Javascript

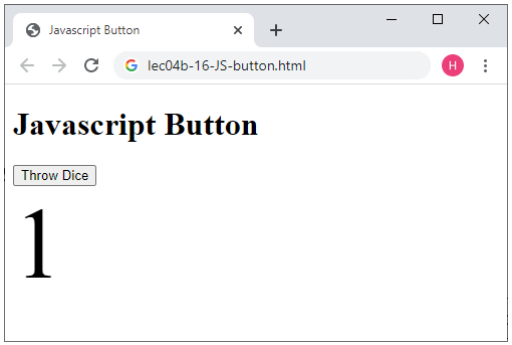
The expression Math.floor(Math.random()*6)+1 will generate a random integer between 1 and 6

The attribute onclick is an event handler that is invoked when this button is clicked. In this case, the Javascript function `throw_dice()` will be called

Javascript built-in function	Scratch Analogy
`Math.floor()`	floor of
`Math.random()`	pick random 0 to 1.0

Math.random() outputs a pseudo-random floating-point number between 0 (inclusive) and 1 (exclusive)

Note that the Scratch block pick random 0 to 1.0 outputs a pseudo-random floating-point number between 0 (inclusive) and 1 (**inclusive**)



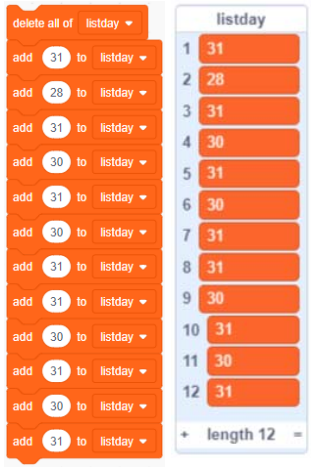
Javascript: Array

- An array can be used to store a list of elements
- Note that the index of the array starts with 0

Javascript array

```
lastday[0]=31
lastday[1]=28
lastday[2]=31
lastday[3]=30
lastday[4]=31
lastday[5]=30
lastday[6]=31
lastday[7]=31
lastday[8]=30
lastday[9]=31
lastday[10]=30
lastday[11]=31
```

Scratch Analogy

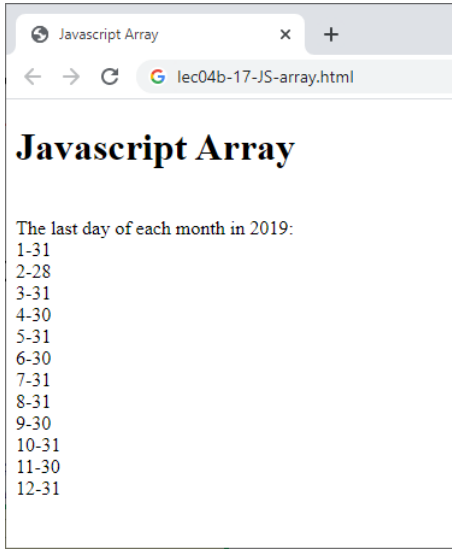


Code Example: lec04b-17-JS-array.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="author" content="Howard Leung" />
    <meta name="description" content="CS1102 Lecture 07b - Javascript Array">
    <meta name="keywords" content="CS1102 Lecture, Javascript, Array" />
    <title>Javascript Array</title>
    <script>

      function init() {
        lastday = [31,28,31,30,31,30,31,31,30,31,30,31];

        for (i=0; i<12; i++) {
          document.getElementById("lastdaylist").innerHTML += (i+1) + "-" + lastday[i] + "<br/>";
        }
      }

    </script>
  </head>
  <body onload="init();">
  <!-- Page content begins here -->
    <h1>Javascript Array</h1>
    <br/>
    The last day of each month in 2018: <br/>
    <div id="lastdaylist" ></div>
    <br/>
  <!-- Page content ends here -->
  </body>
</html>
```

This is one way to initialize the array `lastday` with specific values for 12 elements

The expression `lastday[i]` is used to access the array element with index `i`. Note that the first element of this array is `lastday[0]` and the last element of this array is `lastday[11]`



Interactive Webpage Examples

- Here we provide some specific problems and show you how they can be solved by designing some programs with interactive webpages
- You can try solving the problems yourself before looking at our solution design

Problem 1: List of Numbers and Statistics

- Design a webpage that allows the user to input a list of numbers. The input should be collected one at a time, and validated to make sure that the input is indeed a number. After a valid input is collected, the maximum, minimum, sum and mean among the numbers in the current list should be shown on the webpage.

Building Block 1 for Solving Problem 1: Collecting User Input

- The `prompt` function was one way to collect input from the user
- Here we introduce another way to collect input from the user by using a textbox
  1. In the HTML, the following code can create a textbox:
     ```
     <input type="text" id="textinput" />
     ```
  2. In the HTML, a button can be added to let user click after providing the input in the textbok:
     ```
     <button id="add" onclick="check_and_add();" >Add</button>
     ```
  3. In the Javascript, the content of the textbox can be extracted and stored in a variable `x` by using the following code:
     ```
     var x = document.getElementById("textinput").value;
     ```

1) The user enters 123 in the textbox
2) The user clicks the "Add" button
3) The variable `x` will be assigned with 123 as a string

Building Block 2 for Solving Problem 1: Storing the List of Numbers

- An array will be used to store the list of numbers input by the user
- Let us use the variable name `nlist` to denote this array. As the index of a Javascript array starts with 0, we would like to have
  1. `nlist[0]` stores the 1st number input by the user
  2. `nlist[1]` stores the 2nd number input by the user

  ...
- We need a variable to keep track of how many numbers the user has already input so that we can have the correct index to store each new input. Let us use the variable `N` for this purpose
  - `N` should be initialized to 0
  - Each time when a user provides an input, we first check if the input is indeed a number, and if yes, then we do the followings:

```
nlist[N] = Number(x);
N++;
```

1) `x` is the same variable as defined in previous slide which was assigned the value of the textbox as a string
2) The expression `Number(x)` converts it as a number so that it can be later processed as a number in the calculations
3) The variable `N` should then be incremented by 1

Building Block 3 for Solving Problem 1: Calculating the Statistics (1)

- Calculating the mean is similar to the calculating the sum (by adding each array element) and then dividing by the number of items at the end, i.e.,

```
var sum=0;
for (i=0; i<N; i++)
     sum += nlist[i];
mean = sum/N;
```

- One common way for finding the maximum value in an array is shown as follows:
  1. Initialize a variable called `max` to be a very small number (say the lower bound if known or the smallest number that can be represented)
  2. Compare each element in the array with `max`. If the element is larger than, then assign `max` with this element.

  In other words, the above variable `max` holds the value of the maximum element found so far in the array

  We need to carefully choose the initial value of `max` in order for the above algorithm to work. However, since we know that the array contains at list one element by the time we calculate the maximum value, the above algorithm can be modified as follows:

```
var max=nlist[0];
for (i=1; i<N; i++)
     if (nlist[i] > max)
         max = nlist[i];
```

  With the above, we initialize `max` with the first element in the array, and then compare `max` with each other element and update `max` if a larger value is found

Building Block 3 for Solving Problem 1: Calculating the Statistics (2)

- Finding the minimum value can be achieved in a similar way:

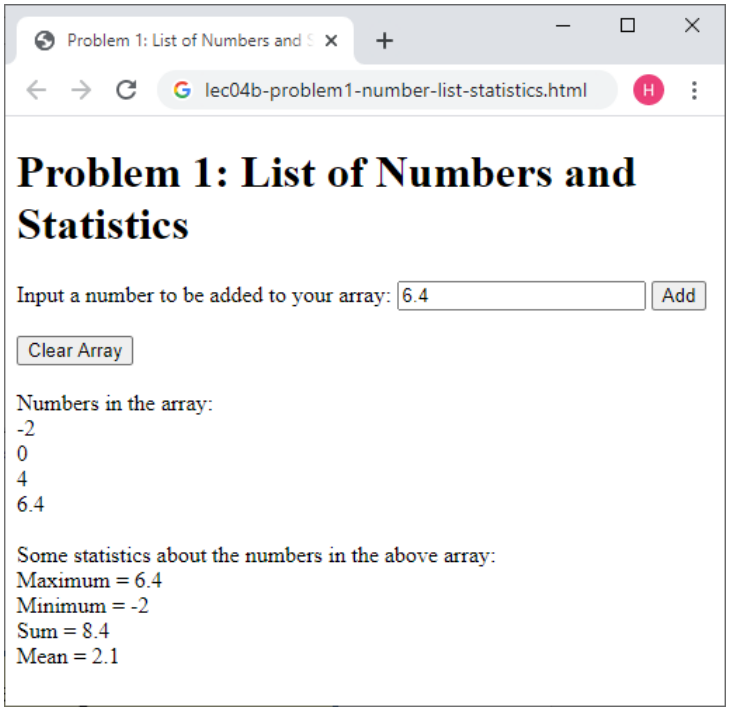
```
var min=nlist[0];
for (i=1; i<N; i++)
     if (nlist[i] < min)
         min = nlist[i];
```

- Finding the sum, mean, maximum and minimum can be done within the same for-loop shown below:

```
var sum=nlist[0];
var max=nlist[0];
var min=nlist[0];
for (i=1; i<N; i++) {
     sum += nlist[i];
     if (nlist[i] > max)
         max = nlist[i];
     if (nlist[i] < min)
         min = nlist[i];
}
mean = sum/N;
```

Solution to Problem 1

```
<!DOCTYPE html>
<html>
  <head>
    <title>Problem 1: List of Numbers and Statistics</title>
    <script>

        var N;
        var nlist = [];

        function clear_array() {
          N = 0;
          document.getElementById("number_list").innerHTML = "";
          document.getElementById("statistics").innerHTML = "";
        }

        function check_and_add() {
          var x = document.getElementById("textinput").value;
          if (isNaN(x)) {
            alert("The input is not a number!");
          }
          else {
            nlist[N] = Number(x);
            N++;

            // Display the newly added item to the end of the list on the webpage
            document.getElementById("number_list").innerHTML += x + "<br/>";

            // Find the maximum, minimum, sum and mean among the numbers in the array
            sum = nlist[0];
            max = nlist[0];
            min = nlist[0];
            for (i=1; i<N; i++) {
              sum += nlist[i];
              if (nlist[i]>max)
                  max = nlist[i];
              if (nlist[i]<min)
                  min = nlist[i];
            }
            mean = sum/N;

            // Display the statistics on the webpage
            document.getElementById("statistics").innerHTML = "Maximum = " + max + "<br/>" + "Minimum = " + min + "<br/>" + "Sum = " + sum + "<br/>" + "Mean = " + mean + "<br/>";

          }
        }

    </script>
  </head>
  <body onload="clear_array();">
  <!-- Page content begins here -->
    <h1>Problem 1: List of Numbers and Statistics</h1>
    Input a number to be added to your array:
    <input type="text" id="textinput" />
    <button id="add" onclick="check_and_add();" >Add</button>
    <br/><br/>
    <button id="clear" onclick="clear_array();" >Clear Array</button>
    <br/><br/>
    Numbers in the array: <br/>
    <div id="number_list" ></div>
    <br/>
    Some statistics about the numbers in the above array: <br/>
    <div id="statistics" ></div>
  <!-- Page content ends here -->
  </body>
</html>
```



Code Example: lec04b-problem1-number-list-statistics.html

Note that one may use the array method `push()` to add a new element to the end of the array and use the array property `length` to get the number of elements in the array. As other programming languages may not have the above method and property defined, this is why we illustrate a more general way to handle array

Problem 2: Letter Counting

- How many F's are there in the following passage?

FINISHED FILES ARE THE RESULT OF YEARS OF SCIENTIFIC STUDY COMBINED WITH THE EXPERIENCE OF YEARS

Building Blocks for Solving Problem 2

- In the HTML, include the passage inside a div element

```
<div id="passage" >
FINISHED FILES ARE THE RESULT OF YEARS OF SCIENTIFIC STUDY COMBINED
WITH THE EXPERIENCE OF YEARS
</div>
```

- In Javascript, extract the passage from the div element and store it in a variable

```
var s = document.getElementById("passage").innerHTML;
```

- Run a loop to iterate over all characters in the string stored in the above variable and have a counting variable (initialized to 0) increased by 1 if the character is 'F'

```
for (count=0, i=0; i<s.length; i++)
    if (s[i] == 'F')
        count++;
```

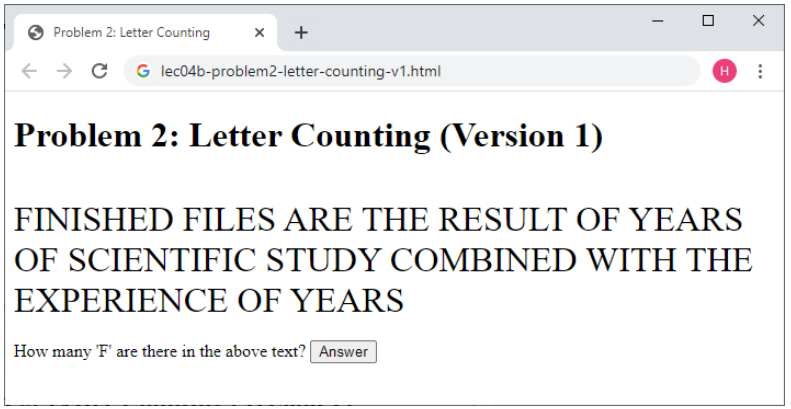
`s.length` returns the length, i.e., the number of characters in the string stored in the variable `s`

Both `count=0` and `i=0` are executed as initialization at the beginning of the for-loop

Note that only a single statement `count++;` needs to be executed when the if-condition is true, so there is no need to use curly brackets `{}` here
Similarly, in the for-loop, only a single statement `if (s[i] == 'F') count++;` needs to be executed at each iteration, so there is also no need to use curly brackets `{}` here

Solution to Problem 2

```
<!DOCTYPE html>
<html>
  <head>
    <title>Problem 2: Letter Counting</title>
    <script>

      function init() {
        document.getElementById("passage").style = "font-size:2em;";
      }

      function show_answer() {
        var s = document.getElementById("passage").innerHTML;

        for (count=0, i=0; i<s.length; i++)
          if (s[i] == 'F')
            count++;

        alert("There are "+count+" F in the text");
      }

    </script>
  </head>
  <body onload="init();">
  <!-- Page content begins here -->
    <h1>Problem 2: Letter Counting (Version 1)</h1>
    <br/>
    <div id="passage" >
      FINISHED FILES ARE THE RESULT OF YEARS OF SCIENTIFIC STUDY COMBINED WITH THE EXPERIENCE OF YEARS
    </div>
    <br/>
    How many 'F' are there in the above text?
    <button id="answer" onclick="show_answer();">Answer</button>
  <!-- Page content ends here -->
  </body>
</html>
```



Code Example: lec04b-problem2-letter-counting-v1.html

Problem 2: Letter Counting (Extra)

- How to add 2 buttons to highlight and unhighlight the F's?

- Design:
  - Right after the webpage is loaded (use `onload` event handler), store the passage specified by the div element in a variable `text`

    ```
    text = document.getElementById("passage").innerHTML;
    ```

  - When the highlight button is clicked, form another variable `newtext` by replacing all occurrences of 'F' in the variable `text` by "F"

    ```
    var newtext = text.replace(/F/g, "<span>F</span>");
    ```

    Replace the div element by this variable `newtext`

    ```
    document.getElementById("passage").innerHTML = newtext;
    ```

    Set the span style to red color

    ```
    span {
        color: red;
    }
    ```

  - When the unhighlight button is clicked, replace the div element by the variable `text` that stores the original passage

    ```
    document.getElementById("passage").innerHTML = text;
    ```

Solution to Problem 2 (Extra)

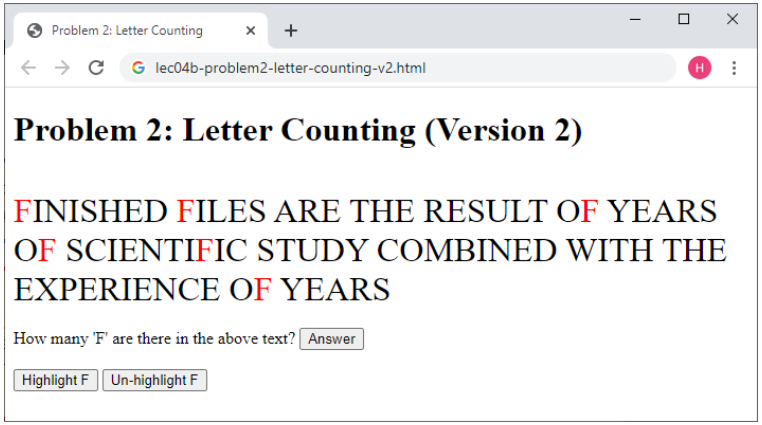
Code Example: lec04b-problem2-letter-counting-v2.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Problem 2: Letter Counting</title>
    <style>
        span {
            color: red;
        }
    </style>
    <script>

        var text;

        function init() {
            document.getElementById("passage").style = "font-size:2em;";

            text = document.getElementById("passage").innerHTML;
        }

        function show_answer() {
            var s = document.getElementById("passage").innerHTML;

            for (count=0, i=0; i<s.length; i++)
                if (s[i] == 'F')
                    count++;

            alert("There are "+count+" F in the text");
        }

        function highlight() {

            var newtext = text.replace(/F/g, "<span>F</span>");

            document.getElementById("passage").innerHTML = newtext;
        }

        function unhighlight() {

            document.getElementById("passage").innerHTML = text;
        }

    </script>
  </head>
  <body onload="init();">
  <!-- Page content begins here -->
    <h1>Problem 2: Letter Counting (Version 2)</h1>
    <br/>
    <div id="passage" >
      FINISHED FILES ARE THE RESULT OF YEARS OF SCIENTIFIC STUDY COMBINED WITH THE EXPERIENCE OF YEARS
    </div>
    <br/>
    How many 'F' are there in the above text?
    <button id="answer" onclick="show_answer();">Answer</button>
    <br/><br/>
    <button id="highlight" onclick="highlight();">Highlight F</button>
    <button id="unhighlight" onclick="unhighlight();">Un-highlight F</button>
  <!-- Page content ends here -->
  </body>
</html>
```

The variable `text` is declared here as a global variable so that all functions can have access (e.g., in functions `init`, `highlight`, `unhighlight`)

Note that if you declare the variable `text` in this function, i.e., if you write `var text = document.getElementById("passage").innerHTML` then it will become a local variable and only this function `init` can have access to it but not others, meaning that the functions `highlight` and `unhighlight` will not see this assigned value when trying to access the variable `text`

The parameter `/F/g` in the function `replace` specifies that all F's are replaced globally. If you change this parameter to `"F"`, then only the first occurrence (instead of all occurrences) will be replaced



Lesson Summary

- Javascript is a programming language that can provide instructions for a browser to dynamically generate content for a website or enhance the website interactivity
- You will not be able to learn everything about HTML, CSS and Javascript after only a few lectures, but it serves as a starting point such that you can explore other features on your own

Reference

[1] Javascript Tutorial

- http://www.w3schools.com/js/