

3

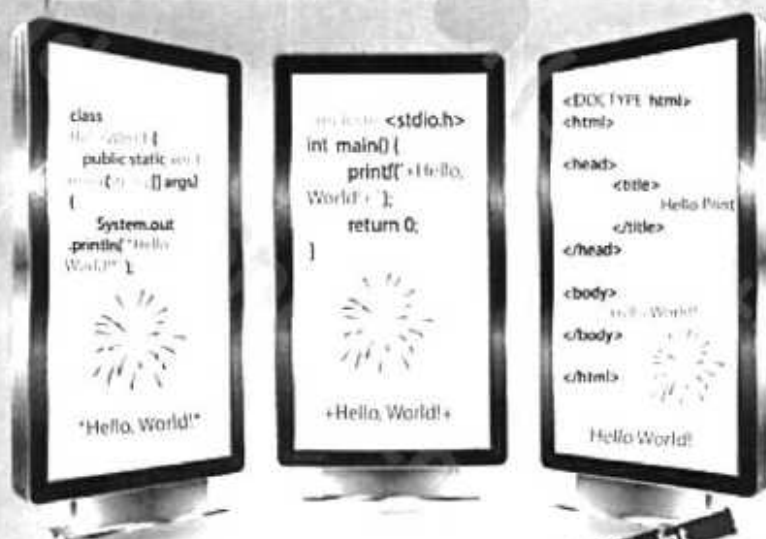
Programming Fundamentals



Learning Outcomes

At the end of this unit students will be able to:

- differentiate between front-end development, and backend development of a website.
- use more advanced HTML/CSS features in an appropriate environment.
- use more advanced programming constructs (lists, etc.) to create dynamic websites using JavaScript as backend scripting
- implement complex algorithms that use more complex data structures (lists, etc.) in JavaScript
- determine more advanced techniques (unit tests, breakpoints, watches) for testing and debugging their code in JavaScript



3.1 Website Development

The development of a website first needs a plan, next tools & resources are chosen & gathered and then it is constructed. It is important to first understand the purpose of the website and the actions you would like visitors to do. Next, you design the navigation layout and appearance of the website. The website is then created using code to accomplish your desired goals. Upon completion, you test the system to ensure proper functioning. The website is finally launched and accessible to everyone.

A website development can be split into two segments, the front-end and the back-end. The most common reasons are:

- First of all, website development & maintenance is easier when the front-end and back-end are kept apart
- Secondly, not everyone is an expert at both. Developers may each concentrate on their areas of expertise because front-end and back-end developers work independently. While back-end developers take care of the technical details like databases and server logic, front-end developers focus on making the website appear attractive and be user-friendly.

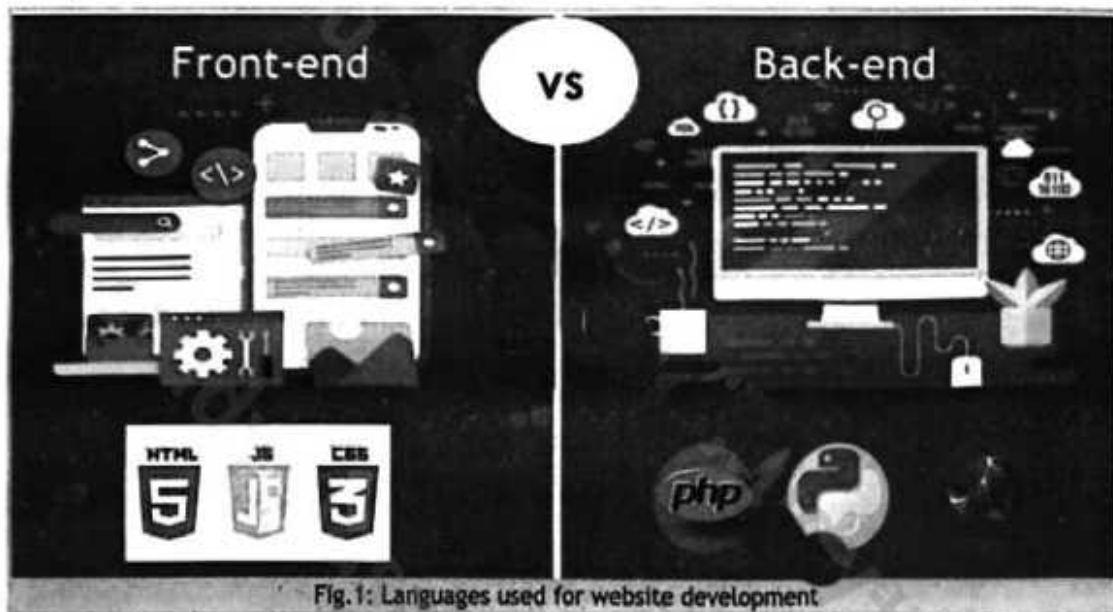


Fig. 1: Languages used for website development

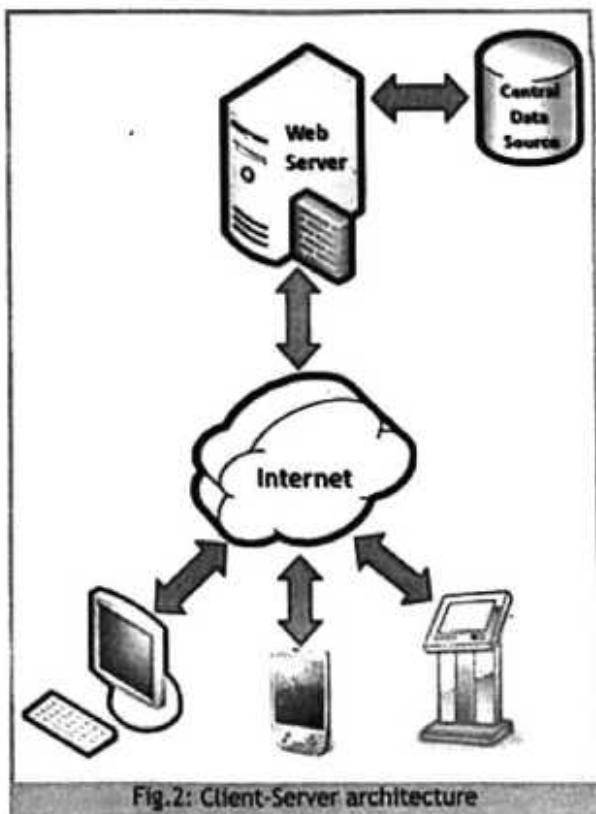
In a website the front-end comprises of everything you see and interact with, such as buttons and images. It is designed to be user-friendly. Front-end developers are responsible for making the website attractive and user-friendly. To do this, they use languages like HTML, CSS, and JavaScript. Front-end is also referred as client-side.

The back-end is where all the data resides and is displayed on-demand basis. Since, the data generally resides on the server, the back-end is also referred as server-side. Back-end developers ensure that everything functions properly behind the website. Back-end developers work on the aspects of the website that you cannot see, such as data management, which they accomplish using a variety of coding languages, like Ruby, Python, php etc, as shown in Fig. 1.

When a user interacts with a webpage, the front-end sends requests to the back-end which processes these requests, interacts with databases and generate dynamic content.

Websites with responsive design adjust their functionality and layout to fit different screen sizes, including those of desktop computers, tablets, and mobile phones. One-size-fits-all website design was the norm in the past, which led to awkward presentations on smaller devices. This problem is solved with responsive design, which makes use of adaptable layouts via which items are scaled in proportion to the screen in terms of percentages rather than fixed pixel values. For example, to improve accessibility on a mobile device, a desktop menu may change to a stacked menu.

For instance, for developing an online store and to run it smoothly both front-end and back-end are required. Shops are created, using the front-end technologies like HTML, CSS and JavaScript, for items to be purchased by the visitors. On the other hand a database comprising of item details, cost and stock, etc. resides on the back-end. When a customer places an order, the order information is sent from the front-end to the back-end. The back-end then checks the data, updates the database and administers the stock.



3.2 Additional Features of HTML/CSS

3.2.1 Forms in HTML

HTML controls how web pages appear and navigate based on user input. HTML may display forms containing text boxes and buttons that users can write or click on like the paperwork is filled out. HTML creates spaces where text may be written, items can be selected from lists and check boxes, etc. To take input on an online form, one very useful element for establishing several types of input fields is the `<input>` element. It makes it easier to take user input in multiple ways in forms such as buttons, checkboxes, text fields, etc.

- A particular kind of `<input>` element called a 'list field' allows users to choose one or more items from a list of already available options. These are frequently utilized for checkbox groups or drop-down menus.
- One more type of `<input>` button is a radio button. It lets the user choose just one possibility from a list of options, which is preferred for choices that are mutually exclusive, like gender.

- Checkboxes are useful for lists of preferences since they allow users to pick multiple choices from a group.
- Button is also used to take input and is frequently used for completing forms, navigating to different sites or execute various functions. The "submit" button is a typical example for transferring information from a form to a server.
- HTML also provides with labels which provide information for user. The <label> element is also used to provide a clear title for each input field.
- The form that is used to gather user input is defined within 'form' tag-pair.

For example, in the following lines of code, on second line a label provides information that student's name needs to be entered in the input field which is created with ID 'studentName'. The input field is created in line 3 so that users may enter the student's name.

```
<form>
  <label for="studentName">Student Name:</label>
  <input type="text" id="studentName" name="studentName"><br><br>
```

With the 'select' tag a drop down list can be created for selecting grade in which the student is. Possible choices of values need to be provided in the next lines so that a drop down list is created for user to choose from the provided choices, only. For instance, to provide choices of 3 classes from 8 to 10, the code should be:

```
<label for="grade">Grade:</label>
<select id="grade" name="grade">
  <option value="1">8</option>
  <option value="2">9</option>
  <option value="3">10</option>
</select><br><br>
```

Mutually exclusive scenarios, where only one among many choices can be true at a time, are catered with radio buttons. To create radio buttons, the input type needs to be set as 'radio' followed by the name for which the radio buttons are set-up and the respective value, for all the choices.

```
<label for="gender">Gender:</label>
<input type="radio" id="male" name="gender" value="male">
<label for="male">Male</label>
<input type="radio" id="female" name="gender" value="female">
<label for="female">Female</label><br><br>
```

On the other hand, for selecting multiple choices check-boxes are used. To create checkboxes, we just need to specify the input type as 'checkbox' followed by ID. Additionally assign the label-name with which the checkbox relates and the value. Thereafter a corresponding text (label) for the user needs to be placed as well. For example, a student can participate in multiple sports so the input can be taken using checkboxes as:

```

<label for="sports">Sports Participated:</label><br>
<input type="checkbox" id="cricket" name="sports" value="cricket">
<label for="cricket">Cricket</label><br>
<input type="checkbox" id="badminton" name="sports" value="badminton">
<label for="badminton">Badminton</label><br>
<input type="checkbox" id="hockey" name="sports" value="hockey">
<label for="hockey">Hockey</label><br>
<input type="checkbox" id="basketball" name="sports" value="basketball">
<label for="basketball">Basketball</label><br><br>

```

The 'for' attribute when used alongside with the <label> element indicates the form element with which a label is tied to. Such input and label needs to be placed for every input choice, like check-boxes.

Lastly, there is a reset button using <input type="reset" value="Reset"> that allows users to click it to clear every field in the form. If all the above codes are gathered in a single file, the output will look like as shown in figure.

OUTPUT:

Student Name: Adil

Grade: 10 ▾

Gender: ☒ Male ☐ Female

Sports Participated:

☒ Cricket

☐ Badminton

☒ Hockey

☐ Basketball

Lets us design a form for visiting the doctor, for the sake of appointment for a check-up. The code is shown in Fig.3.

To gather particulars of the patients and details about the appointments, are as:

- 'form' tag creates a form on line 8.
- The 'action' tells the form where the input data will be submitted, like 'submit_form.html'.
- The 'method' states the way of submission of data, like 'post'.

Note that the different sorts of data are collected with various 'input type'.

- "text" for names,
- "email" for email addresses,
- "tel" is used for phone numbers,
- "date" takes in the selected date and
- "time" for input time.
- "textarea" element creates a large box to write details for the visit.

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <title>Doctor's Visit Form</title>
5  </head>
6  <body>
7    <h1>Doctor's Visit Form</h1>
8    <form action="submit_form.html" method="post"> <p>Patient Information:</p>
9      <label for="name">Name:</label>
10     <input type="text" name="name" id="name" required> <br>
11
12     <input type="radio" id="male" name="Gender" value="Male">
13     <label for="male">Male</label>
14     <input type="radio" id="female" name="Gender" value="Female">
15     <label for="female">Female</label><br><br>
16
17     <input type="checkbox" id="visit1" name="visit1" value="First">
18     <label for="visit1"> First Visit </label>
19     <input type="checkbox" id="visit2" name="visit2" value="Second">
20     <label for="visit2"> Follow up Visit </label>
21     <input type="checkbox" id="visit3" name="visit3" value="Regular">
22     <label for="visit3"> Regular Check-up Visit</label><br> <br>
23
24     <label for="email">Email Address:</label>
25     <input type="email" name="email" id="email" required> <br>
26
27     <label for="phone">Phone Number:</label>
28     <input type="tel" name="phone" id="phone" required> <br>
29
30     <p>Appointment Details:</p>
31     <label for="date">Date of Appointment:</label>
32     <input type="date" name="date" id="date" required> <br>
33
34     <label for="time">Time of Appointment:</label>
35     <input type="time" name="time" id="time" required> <br>
36
37     <p>Reason for Visit:</p>
38     <textarea name="reason" id="reason" rows="5" cols="30" placeholder="
39       Describe your reason for visit here..."></textarea> <br>
40
41     <input type="submit" value="Submit">
42   </form>
43 </body>
44 </html>

```

Fig.3: Code for creating a form- Doctor's visit

Finally, there is a button for submitting the form and a conformation message is shown in the output in Fig.4.

Doctor's Visit Form (a)

Patient Information

Name:

☒ Male ☐ Female

☐ First Visit ☒ Follow-up Visit ☐ Regular Check-up Visit

Email Address:

Phone Number:

Appointment Details:

Date of Appointment:

Time of Appointment:

Reason for Visit:

	10	00	AM
Mild Fever	11	01	PM
Body Pain	12	02	
	01	03	
	02	04	
	03	05	
	04	06	

(b)

Form Submitted !!!

Thank You for Submitting Your Information

Fig.4: Output of form for Doctor's visit



Activity-1

For the output shown in Fig.4, extend the output for 5 records.



Activity-2

Design and create travel reservation form, like that of Fig.4.

3.2.2 Tables in HTML

An easy approach to organize information on webpages is with tables. They work well for presenting information in an easy-to-read and ordered manner. HTML provides option of tables for viewing of data. A table's basic building block, a cell is a rectangular section that can include text, graphics, etc.

The general structure of the table is created by arranging the cells in rows and columns. The horizontal part that represents one line of information within a table and has a number of table cells is called a row while the vertical classification is termed as column.

A header is a specific kind of table cell to name or classify information contained in other cells in the same column. Usually headers are designed to be eye-catching to highlight their significance like bold, underlined, etc.

HTML provides several attributes for creating and modifying tables, with most commonly used tags being:

- <table> tag indicates the beginning and end of a table. Everything inside is a part of the table.
- The <tr> tag creates a row inside the table. Everything between 'tr' tag-pair covers a single row.

The `<th>` tag refers to a table's header cell & contents in `<th>` cells are in bold and centered.

`<td>` tag applies to ordinary table cells. It contains the real data.

• `<caption>` tag puts a heading or description to the table.

• `<thead>`, `<tbody>`, `<tfoot>` tags divide rows into parts, such as headers, major data and summary.

`colspan/rowspan` options allow cells to span several columns or rows, which is convenient for complicated layouts.

In continuation of our example, a sample website with a title is created next in Fig.5, such that a table in HTML is created to display an appointment list. The 'head' section arranges the patient name, date, time, doctor and reason for visit as labels on the top row of the table.

The actual information about appointments is in the 'tbody' section, starting at line 18. There are two sample appointments available right now but we may add more. A 'tr' tag is used for each appointment in a row and a 'td' tag is used for each piece of information inside an appointment. Fig.6 shows the output.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <title>Doctor's Visit Appointments</title>
5  </head>
6  <body>
7    <h1>Doctor's Visit Appointments</h1>
8    <table>
9      <thead>
10     <tr>
11       <th>Patient Name</th>
12       <th>Date</th>
13       <th>Time</th>
14       <th>Doctor</th>
15       <th>Reason for Visit</th>
16     </tr>
17   </thead>
18   <tbody>
19     <tr>
20       <td>Adil</td>
21       <td>2024-05-10</td>
22       <td>10:00 AM</td>
23       <td>Dr. Shams</td>
24       <td>Follow-up</td>
25     </tr>
```

```

26      <tr>
27          <td>Zaki</td>
28          <td>2024-05-15</td>
29          <td>2:00 PM</td>
30          <td>Dr. Qamar</td>
31          <td>Check-up</td>
32      </tr>
33  </tbody>
34 </table>
35 </body>
36 </html>

```

Fig.5: Code to display all appointments

OUTPUT

Doctor's Visit Appointments

Patient Name	Date	Time	Doctor	Reason for Visit
Adil	2024-05-10	10:00 AM	Dr. Shams	Follow-up
Zaki	2024-05-15	2:00 PM	Dr. Qamar	Check-up

Fig.6: List of appointments

Though all information is visible on the website in the table, but it is quite congested and hard to read. So, to tidy up and expand the table for better readability we need to know that the link between the borders of neighbouring table cells is determined by border-collapse as shown in Fig.7. The table becomes neater and more compact when the borders are set to "collapse" which causes them to combine into a single line. On the other hand, when the option is set to "separate," the edges stay distinct and the image seems larger.

```

1  <h4>Table with border-collapse: collapse</h4>
2  <table>
3      <tr>
4          <td>Cell 1</td>
5          <td>Cell 2</td>
6      </tr>
7  </table>
8  <h4>Table with border-collapse: separate</h4>
9  <table border="1" border-collapse="separate">
10     <tr>
11         <td>Cell 1</td>
12         <td>Cell 2</td>
13     </tr>
14 </table>

```

Table with border-collapse: collapse

Cell 1 Cell 2

Table with border-collapse: separate

Cell 1 Cell 2

OUTPUT

Fig.7: Border Collapse

The 'caption' tag provides an overview of the data by placing a caption on top the table, as shown on line 21 in Fig.8.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <title>Doctor's Visit Appointments</title>
5    <style>
6      table {
7        border-collapse: collapse;
8      }
9      th, td {
10       padding: 10px;
11       text-align: left;
12     }
13     th {
14       background-color: gray;
15     }
16   </style>
17 </head>
18 <body>
19   <h1>Doctor's Visit Appointments</h1>
20   <table>
21     <caption>List of upcoming appointments</caption>
22     <thead>
23       <tr>
24         <th>Patient Name</th>
25         <th>Date</th>
26         <th>Time</th>
27         <th>Doctor</th>
28         <th>Reason for Visit</th>
29       </tr>
30     </thead>
31     <tbody>
32       <tr>
33         <td>Adil</td>
34         <td>2024-05-10</td>
35         <td>10:00 AM</td>
36         <td>Dr. Shams</td>
37         <td>Follow-up</td>
38       </tr>
39       <tr>
40         <td>Zaki</td>
```

```

41         <td>2024-05-15</td>
42         <td>2:00 PM</td>
43         <td>Dr. Qamar</td>
44         <td>Check-up</td>
45     </tr>
46 </tbody>
47 </table>
48 </body>
49 </html>

```

Fig.8: Code to display appointments-refined

OUTPUT

Doctor's Visit Appointments

List of upcoming appointments

Patient Name	Date	Time	Doctor	Reason for Visit
Adil	2024-05-10	10:00 AM	Dr Shams	Follow-up
Zaki	2024-05-15	2:00 PM	Dr Qamar	Check-up

Fig.9: Refined layout-list of appointments



Activity-4

Design timetable of your class in html, similar to Fig.9. Add caption 'Timetable for Grade 10'.

Padding is the distance between a cell's border and its content. Padding can be placed to the cell's four sides or specifically to its left, right, bottom or top.

```

1 <table border="1"
  border-collapse="border-collapse">
2   <tr>
3     <td style="padding: 10px">Cell Number
      1</td>
4     <td style="padding: 20px">Cell 2</td>
5     <td style="padding: 5px">Cell 3</td>
6   </tr>
7 </table>

```

Cell Number 1	Cell 2	Cell 3
---------------	--------	--------

Fig.10: Padding

The space among the borders of neighbouring table cells is known as 'cell spacing'. By adjusting the distance between cells, you may make the table appear lengthened.

```

1 <table border="1" cellspacing="20">
2   <tr>
3     <td>Cell Number 1</td>
4     <td>Cell No. 2</td>
5     <td>Cell 3</td>
6   </tr>
7 </table>

```

Cell Number 1	Cell No. 2	Cell 3
---------------	------------	--------

Fig11: Cell Spacing

You may choose the background color of a table, table row, or specific table cell using background color.

```

1  <table border="1" style="background-color:
   □yellow;">
2      <tr>
3          <td>Cell 1</td>
4          <td style="background-color: □cyan;
   ">Cell 2</td>
5          <td>Cell 3</td>
6      </tr>
7  </table>

```

Output

Cell 1	Cell 2	Cell 3
--------	--------	--------

Fig.12: Background Color

To improve the table's appearance, these basic CSS are applied and evident from output in Fig.8.

- 'border-collapse' removes additional lines between cells to make the arrangement look more organized (line-7).
- Padding adds some space around each cell so that cell's text becomes easier to read. (line-10)
- text-align causes the left side of the cell text to align. (line-11)
- To help the table headers prominent, a gray background is provided. (line-14)

Another alternate style for displaying the table is to fill the whole breadth of the available area as highlighted in Fig. 13. A thin line is added around each cell such that each cell in the table stands out clearly, as shown in Fig. 14.

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <title>Doctor's Visit Appointments</title>
5      <style>
6          table {
7              border-collapse: collapse;
8              width: 100%;
9          }
10         th, td {
11             padding: 10px;
12             text-align: left;
13             border: 1px solid □lightgray;
14         }
15         th {
16             background-color: ■gray;
17         }
18     </style>
19 </head>

```

```

20 <body>
21   <h1>Doctor's Visit Appointments</h1>
22   <table cellpadding="0"> <thead>
23     <tr>
24       <th>Patient Name</th>
25       <th>Date</th>
26       <th>Time</th>
27       <th>Doctor</th>
28       <th>Reason for Visit</th>
29     </tr>
30   </thead>
31   <tbody>
32     <tr>
33       <td>Adil</td>
34       <td>2024-05-10</td>
35       <td>10:00 AM</td>
36       <td>Dr. Shams</td>
37       <td>Follow-up</td>
38     </tr>
39     <tr>
40       <td>Zaki</td>
41       <td>2024-05-15</td>
42       <td>2:00 PM</td>
43       <td>Dr. Qamar</td>
44       <td>Check-up</td>
45     </tr>
46   </tbody>
47 </table>
48 </body>
49 </html>

```

Fig.13: Code for list of appointments - Alternate Method

OUTPUT

Patient Name	Date	Time	Doctor	Reason for Visit
Adil	2024-05-10	10:00 AM	Dr. Shams	Follow-up
Zaki	2024-05-15	2:00 PM	Dr. Qamar	Check-up

Fig.14: Alternative layout displaying list of appointments

3.2.3 Animations in CSS

We can easily change the appearance and movement of elements on a webpage via CSS animation like rotate objects or change color. The animation involves creating a number of key-frames that specify the item's styles at different intermediate moments and sequence in the animation. An element's style during animation is specified by the '@keyframes'.

- To represent each keyframe's location in the animation time-line, a percentage value (ranging from 0% to 100%) is provided (line 15-17).
- The <div> tag on line 22 acts as a container that groups and styles other components as a single unit. It is a useful method for structuring components on websites in a logical manner. It is used for layout purposes.
- Formatting the elements of HTML is defined using the <style> tag. The <head> portion of a web page can contain the <style> tag and the matching page elements receive the styles specified in it.

In the code of Fig. 15 we create a basic CSS animation by spinning a red box repeatedly.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <title>Simple Animation</title>
5    <style>
6      #animated-box {
7        margin-top: 50px;
8        margin-left: 50px;
9        width: 100px;
10       height: 100px;
11       background-color: red;
12       animation: spin 2s infinite linear;
13     }
14
15     @keyframes spin {
16       from { transform: rotate(0deg); }
17       to { transform: rotate(360deg); }
18     }
19   </style>
20 </head>
21 <body>
22   <div id="animated-box"></div>
23 </body>
24 </html>
```

Fig. 15: Code to spin a box

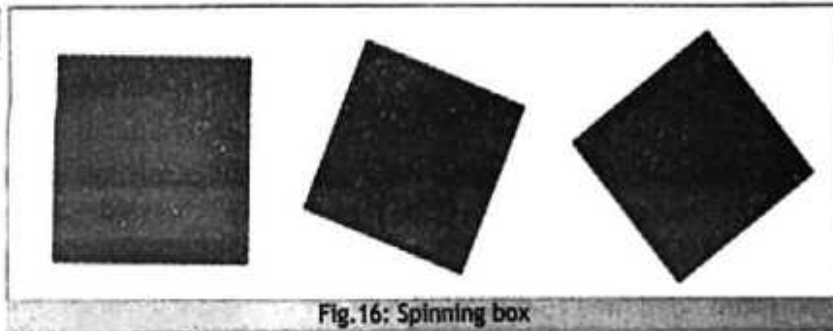


Fig.16: Spinning box

An 'animated-box' ID is created with the 'div' element. Additionally, using inline CSS we directly define the style using 'style' tag in HTML head. We implement a spin animation to the element denoted with the ID "animated-box" in this example as shown in Fig.16. This two-second animation (as specified on line 12) repeats itself indefinitely, where the element's start (0 degrees) and finish points are specified (after executing an entire 360-degree spin).

**Activity-5**

Create a rectangle that changes color from red to orange and then orange to blue, with an interval of 5 seconds.

Another example animation is of changing color of the text in a webpage. The code is shown in Fig.17. This animation changes the color of the text the ID "animated-text" assigned to it. The '@keyframes' rule defines the animation and establishes the order in which color changing take place. The applied animation is called "colorChange" and it animates the text to smoothly change from blue to green every two seconds as shown in Fig. 18.

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <title>Color Change Animation</title>
5    <style>
6      #animated-text {
7        font-size: 20px;
8        color: blue;
9        animation: colorChange 2s infinite alternate;
10     }
11
12     @keyframes colorChange {
13       from { color: blue; }
14       to { color: green; }
15     }
16   </style>
17 </head>
18 <body>
19   <p id="animated-text">This text will change color!</p>
20 </body>
21 </html>

```

Fig.17: Code to change Text color

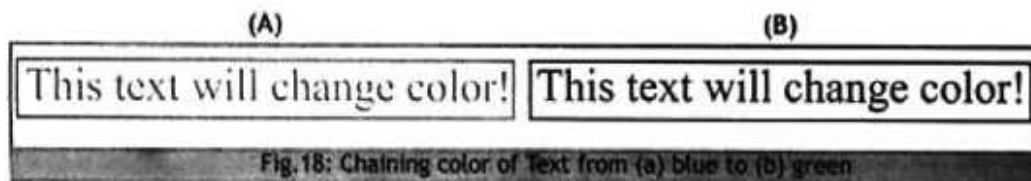


Fig.18: Chaining color of Text from (a) blue to (b) green

We can further extend it to a paragraph where the text in paragraph highlights in various shades of yellow repeatedly, i.e. from lighter to dark shade. This is a simple animation that instead of changing the color text, changes the background color repetitively. Code is listed in Fig.19.

Initially this code creates a paragraph element with the ID "highlighted-text".

Then, to target this specific element, inline CSS styles are applied. An animation with the term "highlight" is used; it repeats endlessly for two seconds.

This animation highlights the text by changing its background color from translucent to yellow.

The "@keyframes" rule is used to outline the actual animation sequence.

The text is initially not highlighted and has a translucent background but is changed to yellow to properly emphasize the content as shown in Fig.20.

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <title>Text Highlighting Animation</title>
5    <style>
6      #highlighted-text {
7        font-size: 20px;
8        color: black;
9        background-color: transparent;
10       animation: highlight 2s ease-in-out alternate
11         infinite;
12     }
13     @keyframes highlight {
14       from { background-color: transparent; }
15       to { background-color: yellow; }
16     }
17   </style>
18 </head>
19 <body>
20   <p id="highlighted-text">This text will be
21     highlighted!</p>
22 </body>
23 </html>

```

Fig.19: Code to change background of P tag iteratively

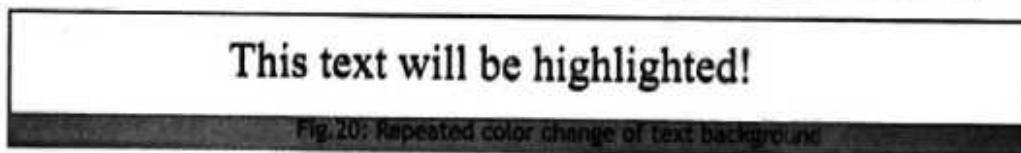


Fig.20: Repeated color change of text background

The whole table including headers and data cells is uniformly styled. Table borders, padding and alignment are managed by properties like 'border', 'padding' and 'text-align'; resulting in a well-organized layout. Every alternate row is given a different background color via the 'nth-child(odd)' rule on line 18. This way all the odd numbered lines will have different background color.

On line 22, the '#footnote' rule applies styles including animation and font size to attribute having the ID "footnote". The "highlight" animation is defined by the '@keyframes highlight' rule on line 27, which changes the footnote's background color from translucent to yellow and back.

Now, let us add the animation style in the Doctor's appointment table in the form of a footnote.

The corresponding code is in Fig.21.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <title>Doctor's Visit Appointments</title>
5 <style>
6   table {
7     border: 2px solid lightgray;
8     border-collapse: collapse;
9   }
10  th, td {
11    padding: 10px;
12    text-align: left;
13  }
14  th {
15    background-color: lightblue;
16    color: white;
17  }
18  tr:nth-child(odd) {
19    background-color: gray;
20  }
21
22  #footnote {
23    font-size: 12px;
24    animation: highlight 2s ease-in-out alternate infinite;
25  }
26
27  @keyframes highlight {
28    from { background-color: transparent; }
29    to { background-color: yellow; }
30  }
31 </style>
```

```

32 </head>
33 <body>
34   <h1>Doctor's Visit Appointments</h1>
35   <table>
36     <caption>List of upcoming appointments</
caption>
37     <thead>
38       <tr>
39         <th>Patient Name</th>
40         <th>Date</th>
41         <th>Time</th>
42         <th>Doctor</th>
43         <th>Reason for Visit</th>
44       </tr>
45     </thead>
46     <tbody>
47       <tr>
48         <td>Adil</td>
49         <td>2024-05-10</td>
50         <td>10:00 AM</td>
51         <td>Dr. Shams</td>
52         <td>Follow-up</td>
53       </tr>
54       <tr>
55         <td>Zaki</td>
56         <td>2024-05-15</td>
57         <td>2:00 PM</td>
58         <td>Dr. Qamar</td>
59         <td>Check-up</td>
60       </tr>
61     </tbody>
62     <tfoot>
63       <tr>
64         <td colspan="5" id="footnote">* The
appointments may subject to change.</td>
65       </tr>
66     </tfoot>
67   </table>
68 </body>
69 </html>

```

Fig. 21: Code to apply animated footnote to a table

Doctor's Visit Appointments

List of upcoming appointments

Adil	2024-05-10	10:00 AM	Dr. Shams	Follow-up
Zaki	2024-05-15	2:00 PM	Dr. Qamar	Check-up
* The appointments may subject to change.				

Fig. 22: Animated footnote for a table

The footnote at the bottom of the table is highlighted as can be seen in Fig.22. “@keyframes highlight” rule as defined on line 27 is used for this animation property. For this a single table cell with the ID “footnote” spans over all the columns on line 64. Such highlighted footnotes are quite useful for projecting important information.

3.3 Advanced Programming Constructs in JavaScript

Conventional websites are static since they are only made with HTML and CSS. Every time a user visits the website, the same content is displayed. This configuration is sufficient for basic informative webpages; however, JavaScript is used to provide further interaction. Few reasons that sophisticated programming concepts is crucial when using JavaScript to create dynamic webpages are as follows:

Interactivity

User input such as clicks, scrolling, and form submissions are not supported by basic HTML. Writing code that responds to these events is made possible by JavaScript, which enhances the user interaction on the website. When a button changes color or a form verifies your data before submitting, such functionalities which manage user interaction are examples of JavaScript enhancement.



Fig. 23: Sample webpage Interactivity via Touchscreen

DATA HANDLING



Fig. 24: Sample data handling in the form of a pie chart

Data Handling

In order to show information about products or user comments, for example, websites frequently need to interact with data. Functions & loops are examples of ideas that help manage and change this data. For instance, you might use functions to carry out computations depending on user input or a loop to list things from a database and plot them in the form a graph .

Dynamic Content Updates

In the absence of JavaScript, modifying the content of a website requires a page refresh. Updates

to some sections of the website may be made without a page reload using JavaScript, which enhance user involvement and performance. For example a newsfeed that updates itself with new content without requiring you to click on a "refresh" button. This way, manual clicks are minimized for newer content updation.

Complex User Interfaces

Drop-down menus, picture slideshows, live chat and other such components are common on modern websites. This can only be accomplished via manipulating the webpage structure, using JavaScript. However at times, if the focus resides more on incorporating more and more features rather than the layout and navigation; it can result in redundancy. Hence, a complex user interface may arise as shown in



Fig. 25: Sample content which are updated on a website

Fig. 26, where every option is accessible from main menu as well as side menu.

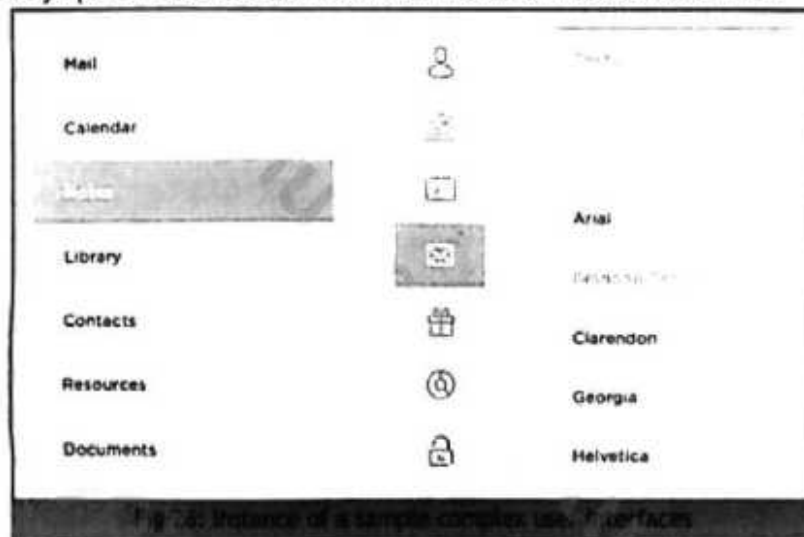


Fig. 26: Instance of a sample complex user interface

JavaScript arrays are a good way for organizing sets of similar data as they allow storing numerous values in a single container. For instance:

Let's say to store names of fruits, distinct variables for every fruit type will look like:

"apple" be fruit1, "banana" be fruit2, "orange" be fruit3, etc.

Alternatively, you may keep every fruit in a single variable by using an array. Creating an array will look like this:

```
const fruits = ["apple", "banana", "orange"];
```

A JavaScript array's index is used to access each element and starts at 0. This indicates that the index of the first element is 0, the index of the second element is 1, and so on. As stated above, our array is named 'fruits' and contains the words "apple", "banana" and "orange". Every object

has an index number where the object is placed in the array. First, "apple" is placed at index 0, followed by "banana" at index 1, then "orange" at index 2. In JavaScript, an array can store various types of data, such as alphabets, numbers or a combination of both.

To traverse elements of an array and if same operation needs to be applied on several components, loops are used. To demonstrate the use of loop that iterates through every item in the 'fruits' array and outputs the result to the console is as follows:

```
1.   for (let i = 0; i < fruits.length; i++)
2.   {
3.     console.log(fruits[i]);
4.   }
```

OUTPUT apple
banana
orange

Key points about arrays:

- You may use 'push()' to add more items to the end and 'pop()' to remove items from the end.
- Use 'length()' to find out how many items are in the array.
- To go through every thing in the array, you can use a 'for-loop'.

JavaScript arrays are flexible in a sense that they can expand or contract to accommodate additions or deletions of items. JavaScript does not require that every element in an array be of the same type, as compared to several other programming languages. Hence, different types of data can be combined in a single array. Arrays are a good tool to manage groups of data in JavaScript code.

Let us start with a simple program to calculate the sum of first 10 positive integers using a for-loop on line 13 as show in Fig.27. Thereafter, it stores the integers in an array using push() function, on line 15 and later prints these numbers and the sum; as can be seen in Fig.28.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>Sum of First 10 Integers</title>
5  </head>
6  <body>
7
8  <script>
9    function calculateSum() {
10      let sum = 0;
11      let numbers = [];
12
13      for (let i = 1; i <= 10; i++) {
14        sum += i;
15        numbers.push(i);
16      }
```

Do you know?

In JavaScript, you can move around an object's structure by using the dot operator (.) for obtaining the parameters and methods of the object. Either to get the value of a property from an object (use the dot operator and then the name of the property) or to call a function.

```

17
18     document.getElementById("sumDisplay").textContent = "The sum of the
19     first 10 integers is: " + sum;
20
21     let output = "The first 10 integers are: [ ";
22     for (let i = 0; i < numbers.length; i++) {
23         output += numbers[i];
24         if (i < numbers.length - 1) {
25             output += ", ";
26         }
27     }
28     output += " ]";
29     document.getElementById("numberDisplay").textContent = output;
30 }
31 </script>
32 <button onclick="calculateSum()">Calculate Sum</button>
33 <br>
34 <p id="numberDisplay"></p>
35 <p id="sumDisplay"></p>
36
37 </body>
38 </html>

```

Fig.27: Program to calculate sum of first 10 positive numbers

On line 9, we define a function namely 'calculateSum()', in which a for-loop is executed starting from 1 till 10. On line 14, a variable 'sum' which has initial value of zero, is added with the index 'i'. This way, by the time the loop terminates, we will be having the sum of all the ten numbers, stored in variable 'sum'. Notable point is the next line where the index is stored in the array using push() function, which will always append (add to the existing) array value of 'i'.

Calculate Sum	OUTPUT
	The first 10 integers are: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
	The sum of the first 10 integers is: 55

Fig.28: Sum of first 10 numbers using array



Activity-3

Change the code of Fig.27 for the following output:

Calculate Sum
The <u>next</u> 10 integers are: [11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
The sum of the <u>next</u> 10 integers is: 155

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <title>Shopping List</title>
5    <style>
6      table {
7        border-collapse: collapse;
8        width: 100%;
9      }
10     th, td {
11       padding: 10px;
12       border: 1px solid lightgray;
13       text-align: left;
14     }
15   </style>
16 </head>
17 <body>
18   <h1>My Shopping List</h1>
19   <input type="text" id="newItem" placeholder="Enter item name">
20   <button onclick="addItem()">Add Item</button>
21   <br>
22   <table id="shoppingListTable"></table>
23
24   <script>
25     let shoppingList = [];
26
27     function addItem() {
28       const newItemName = document.getElementById
29         ("newItem").value;
30
31       if (newItemName !== "") {
32         shoppingList.push(newItemName);
33
34         document.getElementById("newItem").value = "";
35
36         updateShoppingListTable();
37       } else {
38         alert("Please enter an item name!");
39       }
40
41     }
42
43     function updateShoppingListTable() {
44       const table = document.getElementById
45         ("shoppingListTable");

```

```

43
44     table.innerHTML = "";
45
46     const headerRow = document.createElement("tr");
47     const headerItem = document.createElement("th");
48     headerItem.textContent = "Shopping List Items";
49     headerRow.appendChild(headerItem);
50     table.appendChild(headerRow);
51
52     for (let i = 0; i < shoppingList.length; i++) {
53         const itemRow = document.createElement("tr");
54         const itemCell = document.createElement("td");
55         itemCell.textContent = shoppingList[i];
56         itemRow.appendChild(itemCell);
57         table.appendChild(itemRow);
58     }
59 }
60 </script>
61 </body>
62 </html>

```

Fig. 29: Program to add items in a shopping list

OUTPUT

(a)

My Shopping List

Enter item name

(b)

My Shopping List

Enter item name

Shopping List Items

Bread
Butter
Milk
Sugar
Eggs

Fig. 30: Shopping list

The next code of Fig.29 shows how to use a JavaScript array to handle a shopping list. The array keeps track of the item names and the functions in the code handling of adding new items and updating the list on the webpage automatically.

The webpage has a heading, an add-item button, a text box where you may write the products and a table that shows your shopping list.

We maintain the list using JavaScript, as:

- The items are being added in an array we call "shoppingList".
- The "addItem()" function is activated upon clicking the "Add Item" button which checks to see if the box has entered text that is then added to the "shoppingList" and the box is cleared to be rewritten.

- After that the new item is then shown in the table by using a different function called "updateShoppingListTable()".
- Additionally, a top row labeled "Shopping List Items" also is visible.
- Lastly, it adds each item to a new row in the table by going through the "shoppingList" one by one. In this manner, any chosen item to be purchased is added to the table.

The respective output is shown in Fig. 30.

The major functions associated with arrays in JavaScript are:

- **Push()** function adds the value at the end of the array. In the following code, initially an array is created and later another value is inserted using the push() function that appends at the end.

Code:	Output:
<code>oddNumbers = [1, 3, 5, 7, 9];</code>	Initial array: 1,3,5,7,9
<code>oddNumbers.push(11);</code>	After push(11): 1,3,5,7,9,11

- **IndexOf()** function searches for and locates the given element in an array. If present, it returns the index number where the said element resides. For instance:

Code:	Output
<code>oddNumbers.indexOf(11);</code>	Index of last added number (11): 5

- **ToString()** function when applied to an array, converts all the elements of the array into a single string, such that elements are separated by commas. Such as:

Code:	Output:
<code>combinedArray.toString();</code>	1,3,5,7,9,11,2,4,6,8,10

- **Concat()** function merges 2 arrays, like:

Code:	Output:
<code>evenNumbers = [2, 4, 6, 8, 10];</code>	1,3,5,7,9,11,2,4,6,8,10
<code>combinedArray = oddNumbers.concat(evenNumbers);</code>	

- **Slice()** function creates a new subset array of the original array. For example:

Code:	Output:
<code>oddNumbers.slice(2, 4);</code>	Slice of 2 consecutive numbers: 5,7

3.3.2 Bullet Points in HTML

Recall that, you may create lists with bullet points in HTML in two major ways:

- **Unordered List ():**

Lists with bullet points that often resemble squares, circles, or other symbols fall under this category.

- **Ordered List ():**

Lists with bullet points that are numbers (1, 2, 3, etc.) in order fall under this category.

Now, let's modify the doctor appointment example to use an unordered list and incorporate several pricing packages. The code is shown in Fig .31.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <title>Doctor's Visit Appointments</title>
5    <style>
6      table {
7        border: 2px solid black;
8        border-collapse: collapse;
9      }
10     th, td {
11       padding: 10px;
12       text-align: left;
13     }
14     th {
15       background-color: lightblue;
16       color: white;
17     }
18     tr:nth-child(odd) {
19       background-color: lightyellow;
20     }
21   </style>
22 </head>
23 <body>
24   <h1>Doctor's Visit Appointments</h1>
25   <table>
26     <caption>List of upcoming appointments</caption>
27     <thead>
28       <tr>
29         <th>Patient Name</th>
30         <th>Date</th>
31         <th>Time</th>
32         <th>Doctor</th>
33         <th>Reason for Visit</th>
34       </tr>
35     </thead>
36     <tbody>
37       <tr>
38         <td>Adil</td>
39         <td>2024-05-10</td>
40         <td>10:00 AM</td>
41         <td>Dr. Shams</td>
42         <td>Follow-up</td>
43       </tr>
```

```

44     <tr>
45         <td>Zaki</td>
46         <td>2024-05-15</td>
47         <td>2:00 PM</td>
48         <td>Dr. Qamar</td>
49         <td>Check-up</td>
50     </tr>
51 </tbody>
52 <tfoot>
53     <tr>
54         <td colspan="5">
55             * This table displays a sample list of appointments. <br>
56             **Fee Packages:
57             <ul>
58                 <li>Standard Consultation: Rs.500</li>
59                 <li>Follow-up Visit: Rs.300</li>
60                 <li>Comprehensive Check-up: Rs.1,000</li>
61             </ul>
62         </td>
63     </tr>
64 </tfoot>
65 </table>
66 </body>
67 </html>

```

Fig.31: Code to add fee package in bullet points

OUTPUT

Doctor's Visit Appointments

List of upcoming appointments

Adil	2024-05-10	10:00 AM	Dr. Shams	Follow-up
Zaki	2024-05-15	2:00 PM	Dr. Qamar	Check-up

* This table displays a sample list of appointments.
 **Fee Packages:

- Standard Consultation: Rs.500
- Follow-up Visit: Rs.300
- Comprehensive Check-up: Rs.1,000

Fig.32: Fee package in bullet points

Teacher Guide

The colors can either be named or their respective hexadecimal code can be used.

For example, the color name 'LightBlue' can be replaced with #ADD8E6. A detail list can be found at <https://htmlcolorcodes.com>.

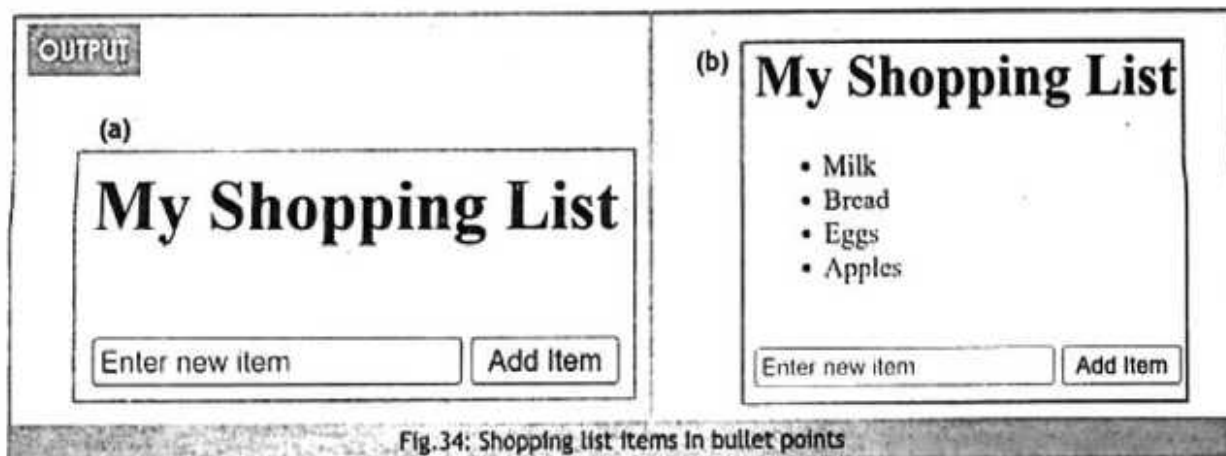
The footnote portion of the page is updated and clearly highlights the fee package, starting from line 52. Earlier code is unchanged. An unordered list (bullet points) may now be found after the footnote on line 57. In this list, each fee package is represented by a list item ("li") element. The line break (
 tag) is inserted between the phrase "Fee Packages:" and the asterisk (*) symbol to make it easier to read and helps in understanding the content, as shown in Fig.32.

3.3.3 Bullet Points can be Values from Arrays

The next example demonstrates how to create dynamic web page features with JavaScript and arrays. The code creates a simple shopping list to which users may add new items. The 'shoppingItems' JavaScript array contains the list of items. The content of the array is used by the `updateList()` method to create list. Any modifications made to the array are therefore reflected in the list. The code is shown in Fig. 33.

```
1  <!DOCTYPE html>
2  <head>
3    <title>Shopping List</title>
4  </head>
5  <body>
6    <h1>My Shopping List</h1>
7    <ul id="shoppingList"></ul>
8  <script>
9    const shoppingItems = ["Milk", "Bread", "Eggs", "Apples"];
10
11  function addItem() {
12    const newItem = document.getElementById("newItemInput").value;
13    shoppingItems.push(newItem);
14    updateList();
15  }
16
17  function updateList() {
18    const listElement = document.getElementById("shoppingList");
19    listElement.innerHTML = "";
20
21    for (let i = 0; i < shoppingItems.length; i++) {
22      const item = shoppingItems[i];
23      const listItem = document.createElement("li");
24      listItem.textContent = item;
25      listElement.appendChild(listItem);
26    }
27  }
28  updateList();
29 </script>
30 <br>
31 <input type="text" id="newItemInput" placeholder="Enter new item">
32 <button onclick="addItem()">Add Item</button>
33 </body>
34 </html>
```

Fig.33: Code to display array contents in bullet points



As can be seen in Fig. 34, the webpage has a heading "My Shopping List", an empty list that displays the items in bullet points and a box where you can enter additional items and an 'add item' button. The "shoppingItems" list defined on line 9 in this code maintains record of all that we add. The "addItem" function is activated upon clicking the "Add Item" button. The text we entered in the box is added to the "shoppingItems" list by this function. The "updateList" method on line 17 is then called in order to ensure that the new item is displayed in our list on the webpage.

A block of code which is repeatedly used for a particular operation is called a function. A function may be defined once and called several times in the program which simplifies code and decreases lines-of-code. When defining a function the 'function' keyword is used followed by the name of the function and parenthesis for any input parameters.

The code for the function is enclosed in curly brackets. The function "updateList" locates every item on the web page and removes any content that is already there. Subsequently, it iteratively moves through the "shoppingItems" list, converting each item to a bullet point and appends it to the list displayed on the website. In this way any purchase is constantly added to the list.

3.4 Complex Algorithms in JavaScript

3.4.1 Array & List are Similar

An ordered sequence of elements is called list. A list is an Abstract Data Type (ADT), where an ADT is the data type which acts for defined operations only. In simple words, though JavaScript provides some basic data types to use but allows to introduce newer data types that are abstract, for which the definition and corresponding operations need to be explicitly mentioned.

Arrays and lists are JavaScript data structures that can hold and manage collections of data. Lists and arrays may both arrange things in an ordered fashion for a set of elements. Every element is stored and accessed via an index number which starts at 0 which shows its position. Based on this index number, values can be searched for and updated if required.

Both are able to hold all types of data, i.e. an array or list can include words, numbers or even a combination of the two. Both are able to expand or contract to hold items, since adding or removing of items as required is quite easy. In simplest form, list is implemented using array.

Major Differences:

- Array is a collection of items, that are stored in memory at contiguous location, whereas list consists of memory locations that are not necessarily contiguous.
- Square brackets [] indicate that an array is a built-in JavaScript tool. Lists are not a component of JavaScript. Tools like arrays are frequently used to create them.
- Since arrays have a fixed arrangement, adding or deleting elements may require some rearranging. Based on how they are constructed, lists can be resized more easily.
- The functions of Arrays are already available, such as to add and remove an item. Lists may need additional steps and are flexible enough to incorporate newer functions and operations. For example, we can define a function 'getElement()' to view the value of current element.
- Arrays are best suited for working with numbers, while lists are chosen for general usage.

Selecting Among Lists and Arrays:

Arrays are a great option for the majority of simple JavaScript tasks since they are simple to use. However, if you want something more versatile, lists are more suitable.

3.4.2 Find an Element in a List

The standard procedure is to search the entire list until you locate the item you're looking for. Here's a basic example that demonstrates how it works. Initially, every item in the "Shopping List" list is shown i.e. bread, apples, milk, and cheese. Thereafter it searches for a certain item (like "milk") and indicates in the "Result" whether or not it is included in the list.

The shopping list's algorithm and how it finds a certain item is as:

Algorithm

1. Initialize:

- (a) To store shopping goods, create an array called shoppingList.
- (b) listContent (string): This is how the shopping list is put together for presentation.
- (c) The item to be searched for is held in itemName (string).

2. Enlisting Shopping List (the initial Loop):

- (a) Iterate shoppingList array using a loop.

3. Within the loop:

- (a) Update the listContent string with the current item.
- (b) If this is not the final item, separate it with a comma and leave space.
- (c) Using the entire listContent string, update the HTML section with the ID "list".

4. The second loop:

- (a) search for an item by going over everything in the shoppingList array.
- (b) Inside loop: Verify that the item we now have matches the itemName that we are searching for.
- (c) If a match is discovered, use return to end the loop and update the HTML section with the ID "result" and a message indicating the item is found.
- (d) Update the HTML code in the section with the ID "result" with a notification indicating the item is not found if the loop concludes without finding a match.

5. Using the Algorithm:

- (a) To begin, call the findItem method and search the required itemName.

The respective implementation of the algorithm is as follows:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <title>Finding Element in Array</title>
5 </head>
6 <body>
7   <h1>Shopping Cart</h1>
8   <p id="list"></p>
9   <p id="result"></p>
10
11 <script>
12   const shoppingList = ["apples", "bread", "milk", "cheese"];
13   function findItem(itemName) {
14     let listContent = "Shopping List: ";
15
16     for (let i = 0; i < shoppingList.length; i++) {
17       listContent += shoppingList[i];
18       if (i < shoppingList.length - 1) {
19         listContent += ", ";
20       }
21     }
22     document.getElementById("list").textContent = listContent;
23     for (let i = 0; i < shoppingList.length; i++) {
24       if (shoppingList[i] === itemName) {
25         document.getElementById("result").textContent = ` "${itemName}"
26           is in your shopping list!`;
27         return;
28       }
29     }
30   }
31 }
```

```

29     document.getElementById("result").textContent = `${itemName} is
      not in your shopping list.`;
30     }
31     findItem("milk");
32 </script>
33 </body>
34 </html>

```

Fig. 35: list-1.html

The program of the above algorithm is shown in Fig.35:

- A shopping-list array with some items is created in line 12.
- On line 13 function 'findItem()' is defined which takes an 'itemName' as an argument and searches for a certain item within the array.
- Next, on line 16 using a for-loop every item in the shopping list is traversed and a string is constructed and printed on line 22.
- Thereafter, the loop on line 23 searches for the respective item, if the item is found then the result is updated as shown in Fig. 36.
- Otherwise a message is displayed on line 29 informing the user that the said item was not found in the list.

OUTPUT

Shopping Cart

Shopping List: apples, bread, milk, cheese

"milk" is in your shopping list!

Fig. 36: list-1.html



Activity-6

Design an algorithm which displays a list of 10 courses offered, and allows user to search for a course.

The webpage output in Fig.36 has undergone minor modifications. There are now two paragraphs.

- First one is the list which displays the whole shopping list. Additionally, a new paragraph displays any search results. We created a new variable named "listContent" on line 14 to do this. All the goods on your shopping list are kept in what looks like a box.
- Next, each item on your list was added to the "listContent" list by means of a loop on lines 16-20. To make it simpler to read, we put commas in between each item. The "list" is updated using its entire "listContent" list after the loop is complete on line 22.

The process of searching for any item remains unchanged. However, we are now searching the shopping list with another loop to find the item in the list.

3.5 Advanced techniques for testing and debugging

3.5.1 Unit Test

In software development, unit test is a crucial activity to ensure that every module of code functions properly, i.e. the code performs as desired. Unit tests are intended to test each line of code to make sure it functions properly. Error detection at an early stage is critical to prevent more serious issues in the later stages.

Unit test also help in outlining function of every section of code. This makes it easier for developers to understand and maintain their code. If the unit tests continue to pass, you may be certain that the primary functionality will continue to function even after you make significant modifications or rearrange the code.

Unit test can be applied to a function, a module or even to a single line of code. For instance, a mathematical formula is to be incorporated to calculate Zakat of yield from an agriculture land. So, the written formula needs to be tested by the developer for different values based on different possible scenarios; till this line of code passes multiple unit tests. Hence, unit tests ensure the quality of code, by providing developers the confidence in making changes, identify errors early and facilitate code management.

Two test cases are presented in Fig.37, for the code of shopping list example of Fig.33. Test 1 looks for the object "milk," which we know exists and anticipates that the function will return "true." Test 2 looks for the nonexistent object "eggs" and anticipates a "false" response from the method. The tests' outcomes are displayed in the Fig.38 based on the return value of the 'findItem' method.

The website layout is mostly unchanged, with a title and a paragraph displaying the unit test results. For the sake of simplicity, the 'findItem()' method is not modified in this example.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <title>Unit Tests for Array Search</title>
5  </head>
6  <body>
7    <h1>Unit Tests for 'findItem' Function</h1>
8    <p id="testResults"></p>
9    <script>
10      const shoppingList = ["apples", "bread", "milk", "cheese"];
11
12      function findItem(itemName) {
13        for (let i = 0; i < shoppingList.length; i++) {
14          if (shoppingList[i] == itemName) {
15            return true;
16          }
17        }
18      }
19    </script>
20  </body>
21 </html>
```

```

17     )
18     return false;
19 }
20
21 let testResults = "";
22 testResults += "Test 1 (Existing Item): ";
23 ✓ if (findItem("milk") == true) {
24     testResults += "Passed \n";
25 } else {
26     testResults += "Failed \n";
27 }
28 testResults += "\n Test 2 (Non-Existing Item): ";
29 if (findItem("eggs") == false) {
30     testResults += "Passed\n";
31 } else {
32     testResults += "Failed\n";
33 }
34 document.getElementById("testResults").textContent = testResults;
35 </script>
36 </body>
37 </html>

```

Fig. 37: Code with unit test

OUTPUT

Unit Tests for 'findItem' Function

Test 1 (Existing Item): Passed

Test 2 (Non-Existing Item): Passed

Fig.38:Unit Test-1.html

3.5.2. Debugging Allows to Analyze the Code

Debugging is useful as:

- Debugging identifies flaws in the code that might affect the layout or functionality of your webpage.
- Debugging allows you to observe step-by-step operation of your code.
- You may create your website more quickly by identifying and rectifying mistakes right away.

Debugging techniques in IDE like Visual Studio.NET involve mainly:

Breakpoint:

Assume you are monitoring a certain line of code where you think that something is dubious. The application pauses at breakpoints and allows you to examine the code (variable values, data flow, etc.).

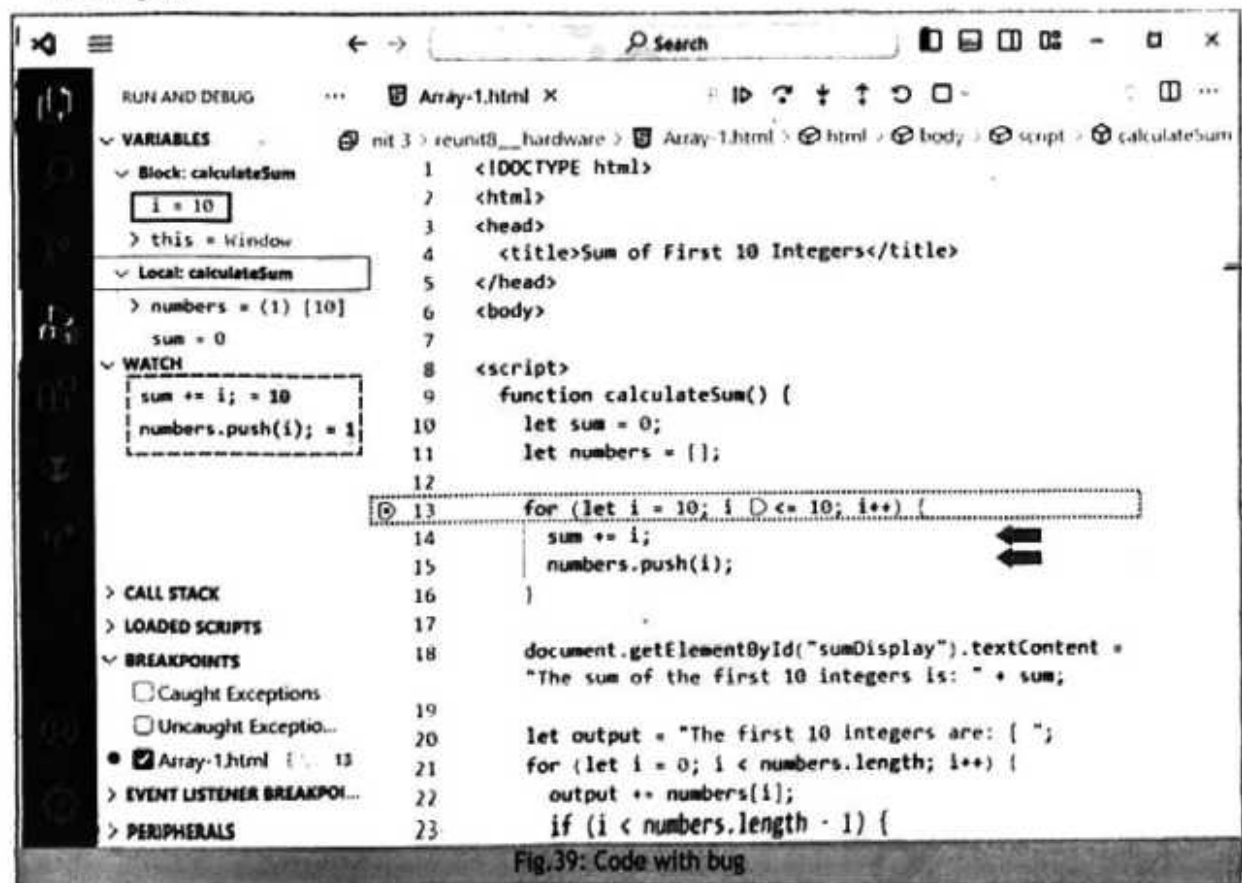
Watchpoint:

In a watchpoint, you establish a rule (for example, whether the value is utilized or changed) on variables or expressions. The software pauses at that particular watchpoint if the rule is followed.

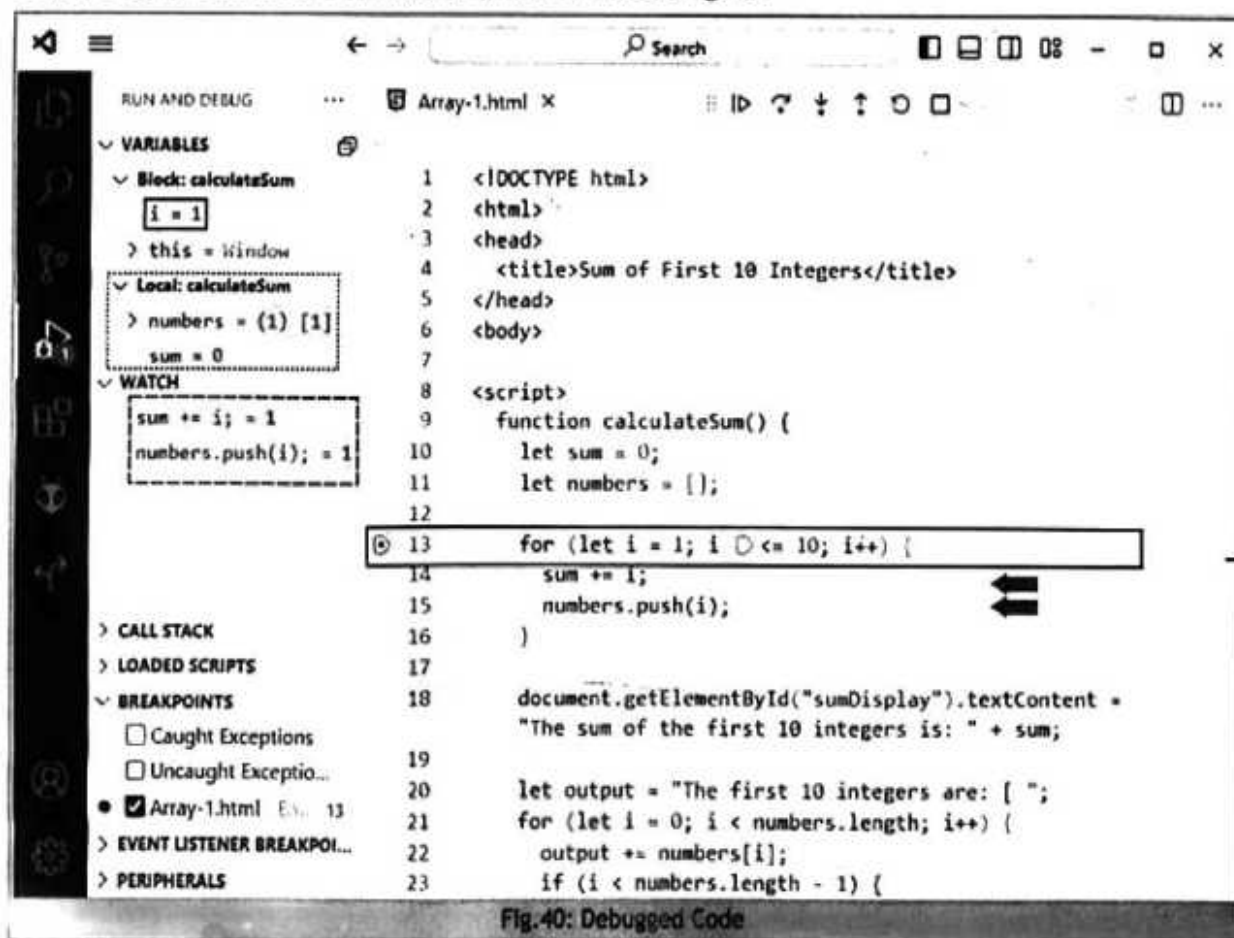
Breakpoints and watchpoints can indicate issues in the code that may be affecting the functionality of your website by highlighting issues and errors.

Let us consider the already discussed example of summing up first 10 integers.

- We have incorporated a bug for the sake of discussion in the for-loop on line 13, in Fig.39. Such things happen unintentionally like typing errors, etc.
- Since, the desired values were not being achieved, so we deployed a breakpoint and added the next 2 lines as watchpoints. Thereafter, we ran the code in debug mode and 'Step-Over' for line by line execution.
- As the loop started, the value of 'i' is shown as 10 in the function block section under variables heading (marked as blue square) and same was the value of 'sum' in watch section (green rectangle).



So, the bug was identified and fixed, as shown in next Fig.40.



Mini Project-1

Write code to take initially 2 arrays with 5 numbers each, as follows:

➤ Arr1 = [1, 3, 5, 7, 9]

➤ Arr2 = [2, 4, 5, 6, 8]

Check every number in the first array against every number in the second array. Display result for each number of the first array whether it was found in the second array or not. Lastly, add 2 unit tests in it by providing the numbers for:

- Check for common elements in both arrays.
- Check for uncommon elements with respect to Arr1, i.e. the numbers which are present in Arr1 but not in Arr2.

Mini Project-2

Create a list such that through buttons, the following operations are executed:

- Create List
- Insert Number
- Find Number
- Remove Number
- Total Numbers in List
- Print List

Whenever a button is pressed, corresponding function should be called.

You may use alerts for user input where necessary, as shown in figure.

List Operations

Create List Insert Number Find Number Remove Number Get Total Numbers Print List

List: 25, 52, 35, 53, 45, 54

Unit Summary

- It is important to first understand the purpose of the website and the actions you would like visitors to do.
- HTML controls how web pages appear and navigate based on user input.
- HTML may display forms containing of text boxes and buttons that users can write or click on like the paperwork is filled out.
- An easy approach to organize information on webpages is with tables. HTML provides option of tables for viewing of data.
- With CSS animations, we can easily change the appearance and movement of elements on a webpage.
- Response to user input such as clicks, scrolling, and form submissions is made possible by JavaScript.
- In the absence of JavaScript, modifying the content of a website requires a page refresh.
- Drop-down menus, picture slideshows, live chat and other such components are common on modern websites. This can only be accomplished via manipulating the webpage structure, using JavaScript.
- In JavaScript, an array stores various types of data, such as alphabets, numbers or a combination of both.
- You may create lists with bullet points in HTML in two major ways, Unordered List and Ordered List.
- Array is a collection of items, that are stored in memory at contiguous location, whereas list consists of memory locations that are not necessarily contiguous.
- Unit tests are intended to test each line of code to make sure it functions properly. Error detection at an early stage is critical to prevent more serious issues in the later stages.
- To monitor a certain line of code, the application pauses at breakpoints and allows you to examine the code (variable values, data flow, etc.).
- In a watchpoint, you establish a rule (for example, whether the value is utilized or changed) on variables or expressions. The software pauses at that particular watchpoint if the rule is followed.

Exercise

Q1. Select the best answer for the following MCQs.

1. Which of the following is primarily associated with front-end development?
 - a) Server management
 - b) Database design
 - c) User interface and design
 - d) Memory Management
2. Which of the following HTML attributes is required in the <input> element for email ?
 - a) type="url"
 - b) type="text"
 - c) type="email"
 - d) type="password"
3. In JavaScript, which method would you use to add a new item to the end of a list?
 - a) push()
 - b) add()
 - c) leftshift()
 - d) rightshift()
4. In JavaScript, how do you access the third element in an array called myArray?
 - a) myArray[3]
 - b) myArray[2]
 - c) myArray[1]
 - d) myArray[0]
5. To manually stop JavaScript code execution at a certain point, you can use:
 - a) console.log()
 - b) stop
 - c) break
 - d) debugger
6. What is the primary purpose of unit tests?
 - a) To validate code syntax
 - b) To test individual parts of the code for correctness
 - c) To style the user interface
 - d) To deploy code to production
7. Front end development is also called_____.
 - a) Label
 - b) Client-side
 - c) Server
 - d) Database
8. Table is an easy way to organize information in_____.
 - a) columns
 - b) rows
 - c) webpages
 - d) list
9. Table's header cell and content is referred in _____ tag.
 - a) <th>
 - b) <td>
 - c) <caption>
 - d) <tr>
10. Link between the borders of neighboring table cell is determined by _____.
 - a) Link border
 - b) border collapse
 - c) border separate
 - d) border

11. Distance between the cell's border and its content is called _____.
 a) Cell b) Output c) Padding d) Style
12. An element style during animation is specified by _____.
 a) @key frame b) @style frame c) @highlighted frame d) @anchor frame
13. A single table cell is indicated by a _____ tag.
 a) <tc> b) <tf> c) <td> d) <th>
14. Arrays and list can contain data types containing _____.
 a) Alphabets b) Numbers c) Alphanumeric d) Checkboxes
15. ADT stands for _____.
 a) Abstract Data type b) Abstract document type
 c) All Data type d) Any Data type

Q2. Give short answers to the following Short Response Questions (SRQs)

1. Give an example where back-end development may not be required and all of the functionalities are handled by front-end.
2. What are the 'td', 'th' and 'tr' tags used for? Explain with the help of an example.
3. Find error(s) in the following codes.

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Bug in Array</title>
5  </head>
6  <body>
7      <h1>Find the Bug</h1>
8      <script>
9          const myArray = ("apple", 3.14,
10             true, "orange");
11          document.write(`<p>My Array: $
12             {myArray[1]}</p>`);
13      </script>
14  </body>
15  </html>

```

(a)

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Bug in the Array</title>
5  </head>
6  <body>
7      <h1>Hunt the Bug</h1>
8      <script>
9          const myArray = "apple", 3.14,
10             true, "orange";
11          document.write(`<p>My Array: $
12             {myArray}</p>`);
13      </script>
14  </body>
15  </html>

```

(b)

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Bug in Array & Loop</
   title>
5 </head>
6 <body>
7   <h1>Numbers are Bugged</h1>
8   <script>
9     const numbers = [1, 2, 3, 4, 5];
10    let output = "";
11    for (i = 0; i <= numbers.length; i
    ++ )
12    {
13      output += numbers[i] + " ";
14    }
15    document.write("<p>" + output +
    "</p>");
16  </script>
17 </body>
18 </html>

```

(c)

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Bugs in Array & Loop</
   title>
5 </head>
6 <body>
7   <h1>Bugs in Numbers </h1>
8   <script>
9     const numbers = [1, 2, 3, 4, 5];
10    let output = "";
11    for (i = numbers.length; i < 0 ;
    i-- )
12    {
13      output += "<p>Number: $
    {numbers[i]}</p>";
14    }
15    document.write(output);
16  </script>
17 </body>
18 </html>

```

(d)

4. How is unit testing used when adding a function to an already written program?
5. Enlist 3 key differences between array and list.
6. How are breakpoints useful in debugging? Elaborate with help of an example.

Q3. Write answers of the following extended response questions in the form of code.

1. For a Bakery, design a simple HTML form to take order online, which includes:

- radio buttons for the option of home delivery or self-collection,
- drop down list for 3 sizes of the cake,
- check boxes for candles, balloons, etc.
- Lastly, add a submit button which generates either an alert or redirects to another form; and responds with an order number.

Sample output is shown as follows:

Order Your Delicious Cake

Delivery or Pickup?

☐ Home Delivery ☐ Self Pickup

Choose Your Cake Size:

Small (Serves 4-6) ▼

Extra Delight (Check all that apply)

- ☐ Chocolate Chips
- ☐ Fresh Strawberries
- ☐ Rainbow Sprinkles

Extras:

- ☐ Birthday Candles
- ☐ Colorful Balloons

[Place Order](#)

Mac//

Your order has been submitted! Order number: 23061

☐ Don't allow this site to request you again

OK

2. Design a website titled "My Weekly Evening Schedule", as follows:

My Weekly Evening Schedule

Day	Activity
Monday	Karate Class
Tuesday	Calligraphy Lessons
Wednesday	Math Test
Thursday	Gardening
Friday	Programming Practice

3. (a) Write code to display the following table:

My Favorite Fruits

Fruit Name	Color
Apple	Red
Banana	Yellow
Orange	Orange

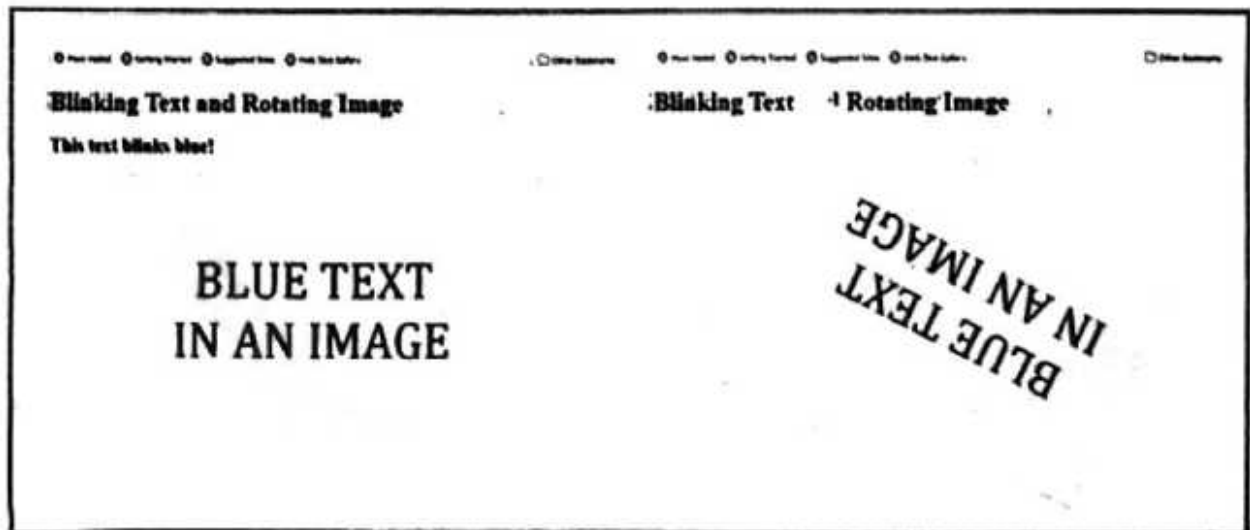
3. (b) Now change the table appearance as follows:

My Favorite Fruits	
Fruit Name	Color
Apple	Red
Banana	Yellow
Orange	Orange

4. Write code for:

- A colored text which repeatedly vanishes for a second and appears in the next second, giving the impression of a “blinking text”
- An image, which spins a whole circle, waits for one second and spins again, in an iterative manner.

Sample output is as follows:



5. Write code to change the bullets into ordered list for the program ‘Sum of first 10 integers’.

6. Create an array with the following values and print it as follows:

`My_Mix_Array : [ apple, 3.14, true, Ali, 25 ]`

7. A list contains 10 entries and we want to search a particular value in the list. Write code, to determine how many iterations it took:

➤➤➤ If the list is found in the list

➤➤➤ If the list is not found in the list