WEB DESIGNING USING PHP
LAB MANUAL

S.R.K.R. ENGINEERING COLLEGE
DEPARTMENT OF INFORMATION TECHNOLOGY
BHIMAVARAM

WEB PAGE DESIGN USING PHP		
(Skill Oriented Course-II)		
Course Objectives: The objectives of the course are to impart:		
1.	Understand the principles of creating an effective web page.	
2.	Understand elements of design with regard to the web.	
3.	Learn the language of the web: HTML and CSS.	
4.	Develop skills in analyzing the usability of a web site.	
5.	Understand how to develop PHP web application with database connectivity.	
6.	Learn CSS grid layout.	
Course outcomes : After completion of the course, students will be able to		**K L**
1	Apply the principles of creating an effective web page.	K3
2	Apply the elements of design with regard to the web.	K3
3	Create the language of the web: HTML and CSS.	K4
4	Develop skills in analyzing the usability of a web site.	K4
5	Understand how to plan and conduct user research related to web usability.	K2
6	Create CSS grid layout	K4

SYLLABUS	
Exercise 1	**Introduction to HTML**
	1.1 What is HTML
	1.2 HTML Documents
	1.3 Basic structure of an HTML document
	1.4 Creating an HTML document
	1.5 Mark up Tags
	1.6 Heading-Paragraphs
	1.7 Line Breaks
	1.8 HTML Tags.
Exercise 2	**Elements of HTML**
	2.1 Introduction to elements of HTML
	2.2 Working with Text
	2.3 Working with Lists, Tables and Frames
	2.4 Working with Hyperlinks, Images and Multimedia
	2.5 Working with Forms and controls.
Exercise 3	**Introduction to Cascading Style Sheets**
	3.1 Concept of CSS
	3.2 Creating Style Sheet
	3.3 CSS Properties
	3.4 CSS Styling(Background, Text Format, Controlling Fonts)
	3.5 Working with block elements and objects

	3.6 Working with Lists and Tables
	3.7 CSS Id and Class
	3.8 Box Model (Introduction, Border properties, Padding Properties, Margin properties)
Exercise 4	4.1 The Basic of JavaScript: Objects,
	4.2 Primitives Operations and Expressions,
	4.3 Screen Output and Keyboard Input,'
	4.4 Object Creation and Modification, Arrays, Functions
	4.5 DHTML: Positioning Moving and Changing Elements
Exercise 5	5.1 Introducing PHP: Creating PHP script,
	5.2 Running PHP script.
	5.3 Using variables,constants,Datatypes, Operators.
	5.4 Conditional statements, Control statements, Arrays ,functions
	5.5 Working with forms and Databases such as MySQL.
	5.6 Develop PHP MySQL CRUD Application

EXERCISE 1

Introduction to HTML

- HTML stands for Hyper Text Markup Language
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

1.1 HTML Documents

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

Output:

My First Heading

My first paragraph.

1.2 Basic structure of an HTML document

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<h1>HAI</h1>
<p>HELLO.</p>
</body>
</html>
```

Output:

HAI

HELLO

1.3 Mark up Tags

Markup Tags

- **HTML**

  This element tells your browser that the file contains HTML coded information. The file extension .html also indicates this is an HTML document and must be used.

- **HEAD**

  Identifies the first part of your HTML-coded document that contains the title.

- **TITLE**

  This element contains your document title and identifies it globally. It is what is displayed as someone's bookmark.

- **BODY**

  The second and largest part of your HTML document, which contains your contents.

1.4 Heading-Paragraphs

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
</body>
</html>
```

Output:

This is a paragraph.

This is another paragraph.

1.5 Line Breaks

PROGRAM:

```
<!DOCTYPE HTML>
1. <html>
2. <head>
3. <title>
4. Example of BR tag
5. </title>
6. </head>
7. <body>
8. <p>If you want to break a line <br> in a paragraph, <br> use the BR element in <br
   > your HTML document. </p>
9. </body>
10. </html>
```

Output:

If you want to break a line
in a paragraph,
use the BR element in
your HTML document.

1.6 HTML Tags.

<!DOCTYPE>	Defines the document type
<html>	Defines an HTML document
<head>	Contains metadata/information for the document
<title>	Defines a title for the document
<body>	Defines the document's body
<h1> to <h6>	Defines HTML headings
<p>	Defines a paragraph
 	Inserts a single line break
<hr>	Defines a thematic change in the content

EXERCISE 2

Elements of HTML

2.1 Introduction to elements of HTML

An HTML element is defined by a start tag, some content, and an end tag.

HTML Elements

The HTML element is everything from the start tag to the end tag:

<tagname>Content goes here...</tagname>

Examples of some HTML elements:

<h1>My First Heading</h1>

My first paragraph.

Start tag	**Element content**	**End tag**
<h1>	My First Heading	</h1>
	My first paragraph.	
 	none	*none*

2.2 Working with Text

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

Output:

My First Heading

My first paragraph.

2.3 Working with Lists, Tables and Frames

TABLES:

Table Cells:

Each table cell is defined by a <td> and a </td> tag.

td stands for table data.

Everything between <td> and </td> are the content of the table cell.

Table Rows:

Each table row starts with a <tr> and end with a </tr> tag.

tr stands for table row.

Table Headers:

Sometimes you want your cells to be headers, in those cases use the <th> tag instead of the <td> tag:

HTML Table Borders

When you add a border to a table, you also add borders around each table cell:

To add a border, use the CSS border property on table, th, and td elements:

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
}
</style>
```

```
</head>
<body>
<table style="width:100%">
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Age</th>
</tr>
<tr>
<td>Jill</td>
<td>Smith</td>
<td>50</td>
</tr>
<tr>
<td>Eve</td>
<td>Jackson</td>
<td>94</td>
</tr>
<tr>
<td>John</td>
<td>Doe</td>
<td>80</td>
</tr>
</table>
</body>
</html>
```

Output:

Firstname	Lastname	Age
Jill	Smith	50

Eve	Jackson	94
John	Doe	80

Collapsed Table Borders:

To avoid having double borders like in the example above, set the CSS border-collapse property to collapse.

This will make the borders collapse into a single border:

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
</style>
</head>
<body>
<table style="width:100%">
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Age</th>
```

```
</tr>
<tr>
<td>Jill</td>
<td>Smith</td>
<td>50</td>
</tr>
<tr>
<td>Eve</td>
<td>Jackson</td>
<td>94</td>
</tr>
<tr>
<td>John</td>
<td>Doe</td>
<td>80</td>
</tr>
</table>
</body>
</html>
```

Output:

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

Style Table Borders:

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
```

```
<style>
table, th, td {
  border: 1px solid white;
  border-collapse: collapse;
}
th, td {
  background-color: #96D4D4;
}
</style>
</head>
<body>
<table style="width:100%">
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Age</th>
</tr>
<tr>
<td>Jill</td>
<td>Smith</td>
<td>50</td>
</tr>
<tr>
<td>Eve</td>
<td>Jackson</td>
<td>94</td>
</tr>
<tr>
<td>John</td>
```

```
<td>Doe</td>

<td>80</td>

</tr>

</table>

</body>

</html>
```

Output:

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

HTML Table Padding & Spacing

HTML tables can adjust the padding inside the cells, and also the space between the cells.

With Padding

hello	hello	hello
hello	hello	hello
hello	hello	hello

With Spacing

hello	hello	hello
hello	hello	hello
hello	hello	hello

HTML Table - Cell Padding

Cell padding is the space between the cell edges and the cell content.

By default the padding is set to 0.

To add padding on table cells, use the CSS padding property:

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
th, td {
  padding: 15px;
}
</style>
</head>
<body>
<h2>Cellpadding</h2>
<table style="width:100%">
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Age</th>
</tr>
<tr>
<td>Jill</td>
<td>Smith</td>
```

```
<td>50</td>
</tr>
<tr>
<td>Eve</td>
<td>Jackson</td>
<td>94</td>
</tr>
<tr>
<td>John</td>
<td>Doe</td>
<td>80</td>
</tr>
</table>
<p><strong>Tip:</strong> Try to change the padding to 5px.</p>
</body>
</html>
```

- To add padding only above the content, use the padding-top property.
- And the others sides with the padding-bottom, padding-left, and padding-right properties:

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
th, td {
  padding-top: 10px;
```

```
  padding-bottom: 20px;
  padding-left: 30px;
  padding-right: 40px;
}
</style>
</head>
<body>
<h2>Cellpadding - top - bottom - left - right </h2>
<table style="width:100%">
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Age</th>
</tr>
<tr>
<td>Jill</td>
<td>Smith</td>
<td>50</td>
</tr>
<tr>
<td>Eve</td>
<td>Jackson</td>
<td>94</td>
</tr>
<tr>
<td>John</td>
<td>Doe</td>
<td>80</td>
</tr>
```

</table>

</body>

</html>

Output:

Cellpadding - top - bottom - left - right

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

- **HTML Table - Cell Spacing**

Cell spacing is the space between each cell.

By default the space is set to 2 pixels.

To change the space between table cells, use the CSS border-spacing property on the table element:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
}
table {
  border-spacing: 30px;
}
```

```
</style>
</head>
<body>
<h2>Cellspacing</h2>
<table style="width:100%">
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Age</th>
</tr>
<tr>
<td>Jill</td>
<td>Smith</td>
<td>50</td>
</tr>
<tr>
<td>Eve</td>
<td>Jackson</td>
<td>94</td>
</tr>
<tr>
<td>John</td>
<td>Doe</td>
<td>80</td>
</tr>
</table>
</body>
</html>
```

Output:

Cellspacing

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

HTML Table Colspan&Rowspan

HTML tables can have cells that spans over multiple rows and/or columns.

NAME		

APRIL		

HTML Table - Colspan

To make a cell span over multiple columns, use the colspan attribute

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
```

```
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
</style>
</head>
<body>
<h2>Cell that spans two columns</h2>
<p>To make a cell span more than one column, use the colspanattribute.</p>
<table style="width:100%">
<tr>
<thcolspan="2">Name</th>
<th>Age</th>
</tr>
<tr>
<td>Jill</td>
<td>Smith</td>
<td>43</td>
</tr>
<tr>
<td>Eve</td>
<td>Jackson</td>
<td>57</td>
</tr>
</table>
</body>
</html>
```

Output:

HTML Table - ColspanTo make a cell span over multiple columns, use the colspan attribute

Cell that spans two columns

To make a cell span more than one column, use the colspan attribute.

Name		Age
Jill	Smith	43
Eve	Jackson	57

HTML Table - Rowspan

To make a cell span over multiple rows, use the rowspan attribute:

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
</style>
</head>
<body>
<h2>Cell that spans two rows</h2>
<p>To make a cell span more than one row, use the rowspanattribute.</p>
<table style="width:100%">
<tr>
<th>Name</th>
<td>Jill</td>
```

```
</tr>
<tr>
<throwspan="2">Phone</th>
<td>555-1234</td>
</tr>
<tr>
<td>555-8745</td>
</tr>
</table>
</body>
</html>
```

Output:

Cell that spans two rows

To make a cell span more than one row, use the rowspan attribute.

Name	Jill
Phone	555-1234
	555-8745

HTML Table Colgroup

The <colgroup> element is used to style specific columns of a table.

HTML Table Colgroup

If you want to style the two first columns of a table, use the <colgroup> and <col> elements.

MON	TUE	WED	THU	FRI	SAT	SUN
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21

22	23	24	25	26	27	28

The <colgroup> element should be used as a container for the column specifications.

Each group are specified with a <col> element.

The span attribute specifies how many columns that gets the style.

The style attribute specifies the style to give the columns.

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
</style>
</head>
<body>
<h2>Colgroup</h2>
<table style="width: 100%;">
<colgroup>
<col span="2" style="background-color: #D6EEEE">
</colgroup>
<tr>
<th>MON</th>
<th>TUE</th>
<th>WED</th>
<th>THU</th>
```

```
<th>FRI</th>
<th>SAT</th>
<th>SUN</th>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>24</td
</tr>
</table>
</body>
</html>
```

Output:

Colgroup

MON	TUE	WED	THU	FRI	SAT	SUN
1	2	3				
8	9	10				
15	16	17				
22	23	24				

HTML LIST TAGS:

	Defines an unordered list
	Defines an ordered list
	Defines a list item
<dl>	Defines a description list
<dt>	Defines a term in a description list
<dd>	Describes the term in a description list

HTML Unordered Lists

The HTML <ul> tag defines an unordered (bulleted) list.

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<h2>An unordered HTML list</h2>
<ul>
<li>Coffee</li>
<li>Tea</li>
<li>Milk</li>
</ul>
</body>
</html>
```

Output:

An unordered HTML list

- Coffee
- Tea
- Milk

Ordered HTML List

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.

The list items will be marked with numbers by default:

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<h2>An ordered HTML list</h2>
<ol>
```

```
<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

</body>

</html>
```

Output:

An ordered HTML list

1. Coffee
2. Tea
3. Milk

HTML Description Lists

The <dl> tag defines the description list, the <dt> tag defines the term (name), and the <dd> tag describes each term:

PROGRAM:

```
<!DOCTYPE html>

<html>

<body>

<h2>A Description List</h2>

<dl>

<dt>Coffee</dt>

<dd>- black hot drink</dd>

<dt>Milk</dt>

<dd>- white cold drink</dd>

</dl>

</body>

</html>
```

Output:

A Description List

Coffee

- black hot drink

Milk

- white cold drink

FRAMES:

HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document.

A collection of frames in the browser window is known as a frameset.

The window is divided into frames in a similar way the tables are organized: into rows and columns.

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Frames</title>
</head>
<frameset rows = "10%,80%,10%">
<frame name = "top" src = "/html/top_frame.htm" />
<frame name = "main" src = "/html/main_frame.htm" />
<frame name = "bottom" src = "/html/bottom_frame.htm" />
<noframes>
<body>Your browser does not support frames.</body>
</noframes>
</frameset>
</html>
```

PROGRAM:

```
<html>
<head>
<title>HTML Frames</title>
</head>
<framesetcols="25%,50%,25%">
<framename="left"src="/html/top_frame.htm"/>
<framename="center"src="/html/main_frame.htm"/>
<framename="right"src="/html/bottom_frame.htm"/>
<noframes>
<body>Your browser does not support frames.</body>
</noframes>
</frameset>
</html>
```

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Frames</title>
</head>
<frameset rows = "30%,20%,*">
<frame src = "/html/top_frame.htm" />
<frame src = "/html/main_frame.htm" />
<framesetcols="50%,*">
<framesrc="/html/top_frame.htm"/>
<framesrc="/html/main_frame.htm"/>
</frameset>
</frameset>
</html>
```

2.4 Working with Hyperlinks, Images and Multimedia

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<h2>HTML Image</h2>
<imgsrc="pic_trulli.jpg" alt="Trulli" width="500" height="333">
</body>
</html>
<!DOCTYPE html>
<html>
<body>
<h1>HTML Links</h1>
<p><a href="https://www.w3schools.com/">Visit W3Schools.com!</a></p>
</body>
</html>
```

Output:

HTML Links

Visit W3Schools.com!

2.5 Working with Forms and controls.

PROGRAM:

```
<!DOCTYPE html>

<html>

<head>

<title>Simple registration form</title>

<link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700"
rel="stylesheet">

<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.4.1/css/all.css"
integrity="sha384-
5sAR7xN1Nv6T6+dT2mhtzEpVJvfS3NScPQTrOxhwjIuvcA67KV2R5Jz6kr4abQsz"
crossorigin="anonymous">

<style>

    html, body {

    display: flex;

    justify-content: center;

    height: 100%;

    }

    body, div, h1, form, input, p {

    padding: 0;

    margin: 0;

    outline: none;

    font-family: Roboto, Arial, sans-serif;

    font-size: 16px;

color: #666;

    }

    h1 {

    padding: 10px 0;

    font-size: 32px;

    font-weight: 300;
```

```
    text-align: center;
    }
    p {
    font-size: 12px;
    }
    hr {
color: #a9a9a9;
    opacity: 0.3;
    }
.main-block {
    max-width: 340px;
    min-height: 460px;
    padding: 10px 0;
    margin: auto;
    border-radius: 5px;
    border: solid 1px #ccc;
    box-shadow: 1px 2px 5px rgba(0,0,0,.31);
    background: #ebebeb;
    }
    form {
    margin: 0 30px;
    }
.account-type, .gender {
    margin: 15px 0;
    }
    input[type=radio] {
    display: none;
    }
label#icon {
```

```
    margin: 0;
    border-radius: 5px 0 0 5px;
    }
label.radio {
    position: relative;
    display: inline-block;
    padding-top: 4px;
    margin-right: 20px;
    text-indent: 30px;
    overflow: visible;
    cursor: pointer;
    }
label.radio:before {
    content: "";
    position: absolute;
    top: 2px;
    left: 0;
    width: 20px;
    height: 20px;
    border-radius: 50%;
    background: #1c87c9;
    }
label.radio:after {
    content: "";
    position: absolute;
    width: 9px;
    height: 4px;
    top: 8px;
    left: 4px;
```

```
    border: 3px solid #fff;
    border-top: none;
    border-right: none;
    transform: rotate(-45deg);
    opacity: 0;
    }
    input[type=radio]:checked + label:after {
    opacity: 1;
    }
    input[type=text], input[type=password] {
    width: calc(100% - 57px);
    height: 36px;
    margin: 13px 0 0 -5px;
    padding-left: 10px;
    border-radius: 0 5px 5px 0;
    border: solid 1px #cbc9c9;
    box-shadow: 1px 2px 5px rgba(0,0,0,.09);
    background: #fff;
    }
    input[type=password] {
    margin-bottom: 15px;
    }
    #icon {
    display: inline-block;
    padding: 9.3px 15px;
    box-shadow: 1px 2px 5px rgba(0,0,0,.09);
    background: #1c87c9;
color: #fff;
    text-align: center;
```

```
    }
.btn-block {
    margin-top: 10px;
    text-align: center;
    }
    button {
    width: 100%;
    padding: 10px 0;
    margin: 10px auto;
    border-radius: 5px;
    border: none;
    background: #1c87c9;
    font-size: 14px;
    font-weight: 600;
color: #fff;
    }
button:hover {
    background: #26a9e0;
    }
</style>
</head>
<body>
<div class="main-block">
<h1>Registration</h1>
<form action="/">
<hr>
<div class="account-type">
<input type="radio" value="none" id="radioOne" name="account" checked/>
<label for="radioOne" class="radio">Personal</label>
```

```
<input type="radio" value="none" id="radioTwo" name="account" />
<label for="radioTwo" class="radio">Company</label>
</div>
<hr>
<label id="icon" for="name"><i class="fas fa-envelope"></i></label>
<input type="text" name="name" id="name" placeholder="Email" required/>
<label id="icon" for="name"><i class="fas fa-user"></i></label>
<input type="text" name="name" id="name" placeholder="Name" required/>
<label id="icon" for="name"><i class="fas fa-unlock-alt"></i></label>
<input type="password" name="name" id="name" placeholder="Password" required/>
<hr>
<div class="gender">
<input type="radio" value="none" id="male" name="gender" checked/>
<label for="male" class="radio">Male</label>
<input type="radio" value="none" id="female" name="gender" />
<label for="female" class="radio">Female</label>
</div>
<hr>
<div class="btn-block">
<p>By clicking Register, you agree on our <a href="https://www.w3docs.com/privacy-policy">Privacy Policy for W3Docs</a>.</p>
<button type="submit" href="/">Submit</button>
</div>
</form>
</div>
</body>
```

Output:


Registration
Personal Company
Male Female
By clicking Register, you agree on our Privacy Policy for W3Docs.
Submit

EXERCISE 3

Introduction to Cascading Style Sheets

3.1 Concept of CSS

Cascading Style Sheets (CSS) is used to format the layout of a webpage.

With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colors are to be used, different displays for different devices and screen sizes, and much more!

Using CSS

CSS can be added to HTML documents in 3 ways:

- **Inline** - by using the style attribute inside HTML elements
- **Internal** - by using a <style> element in the <head> section
- **External** - by using a <link> element to link to an external CSS file

Inline CSS:

- An inline CSS is used to apply a unique style to a single HTML element.
- An inline CSS uses the style attribute of an HTML element.
- The following example sets the text color of the <h1> element to blue, and the text color of the <p> element to red:

Internal CSS:

- An internal CSS is used to define a style for a single HTML page.
- An internal CSS is defined in the <head> section of an HTML page, within a <style> element.

External CSS:

- An external style sheet is used to define the style for many HTML pages.
- To use an external style sheet, add a link to it in the <head> section of each HTML page.

3.2 Creating Style Sheet

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
<title>Inline CSS</title>
</head>
<body>
<p style = "color:#009900; font-size:50px;
font-style:italic; text-align:center;">
GeeksForGeeks
</p>
</body>
</html>
```

Output:

GeeksForGeeks

3.3 CSS Properties

The CSS color property defines the text color to be used.

The CSS font-family property defines the font to be used.

The CSS font-size property defines the text size to be used.

The CSS border property defines a border around an HTML element.

The CSS margin property defines a margin (space) outside the border.

The CSS padding property defines a padding (space) between the text and the border

Program:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  color: blue;
  font-family: verdana;
  font-size: 300%;
  text-align:center;
  border: 2px solid black;
  padding: 50px;
  margin:100px;
}
p  {
  color: red;
  font-family: courier;
  font-size: 160%;
  text-align:center;
  border: 2px solid black;
  padding: 20px;
  margin:20px;

}
```

```
</style>

</head>

<body>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

</body>

</html>
```

Output:

3.4 CSS Styling(Background, Text Format, Controlling Fonts)

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: lightblue;
}
</style>
</head>
<body>
<h1>Hello World!</h1>
<p>This page has a light blue background color!</p>
</body>
</html>
```

Output:

Hello World!

This page has a light blue background color!

Program:

```
<!DOCTYPE html>
<html>
```

```
<head>
<style>
.p1 {
  font-family: "Times New Roman", Times, serif;
}
.p2 {
  font-family: Arial, Helvetica, sans-serif;
}
.p3 {
  font-family: "Lucida Console", "Courier New", monospace;
}
</style>
</head>
<body>
<h1>CSS font-family</h1>
<p class="p1">This is a paragraph, shown in the Times New Roman font.</p>
<p class="p2">This is a paragraph, shown in the Arial font.</p>
<p class="p3">This is a paragraph, shown in the Lucida Console font.</p>
</body>
</html>
```

Output:

CSS font-family

This is a paragraph, shown in the Times New Roman font.

This is a paragraph, shown in the Arial font.

This is a paragraph, shown in the Lucida Console font.

3.5 Working with block elements and objects

HTML element has a default display value, depending on what type of element it is.

There are two display values: block and inline.

Block-level Elements

A block-level element always starts on a new line, and the browsers automatically add some space (a margin) before and after the element.

A block-level element always takes up the full width available (stretches out to the left and right as far as it can).

Two commonly used block elements are: <p> and .

The <p> element defines a paragraph in an HTML document.

The element defines a division or a section in an HTML document.

PROGRAM:

```
<!DOCTYPE html>

<html>

<body>

<p style="border: 1px solid black">Hello World</p>

<div style="border: 1px solid black">Hello World</div>


<p>The P and the DIV elements are both block elements, and they will always start on a new line and take up the full width available (stretches out to the left and right as far as it can).</p>

</body>

</html>
```

Output:

Hello World

Hello World

The P and the DIV elements are both block elements, and they will always start on a new line and take up the full width available (stretches out to the left and right as far as it can).

3.6 Working with Lists and Tables

PROGRAM:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
}
</style>
</head>
<body>
<table style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
```

```
  <td>Eve</td>
  <td>Jackson</td>
  <td>94</td>
 </tr>
 <tr>
  <td>John</td>
  <td>Doe</td>
  <td>80</td>
 </tr>
</table>
</body>
</html>
```

Output:

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

Program:

```
<!DOCTYPE html>

<html>

<body>

<h2>Unordered List with Disc Bullets</h2>

<ul style="list-style-type:disc;">

  <li>Coffee</li>

  <li>Tea</li>

  <li>Milk</li>

</ul>

</body>

</html>
```

Output:

Unordered List with Disc Bullets

- Coffee
- Tea
- Milk

3.7 CSS Id and Class

PROGRAM:

```
<!DOCTYPE html>

<html>


<head>
   <title>
      HTML id attribute
   </title>


   <style>
      #geeks{
         color: green;
         font-size: 25px;
      }
   </style>
</head>

<body style="text-align: center">
   <h1>Geeks for Geeks</h1>
   <p id="geeks">Welcome to Geeks for Geeks</p>
   <p>A Computer Science portal for geeks</p>
</body>

</html>
```

Output:

Geeks for Geeks

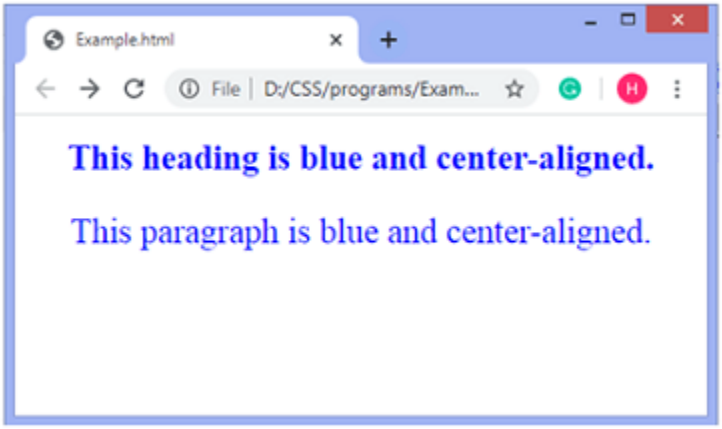
Welcome to Geeks for Geeks

A Computer Science portal for geeks

PROGRAM:

```
1.  !DOCTYPE html>
2.  <html>
3.  <head>
4.  <style>
5.  .example {
6.  text-align: center;
7.  color: blue;
8.  font-size: 25px;
9.  }
10. </style>
11. </head>
12. <body>
13. <h1 class="example">This heading is blue and center-aligned.</h1>
14. <p class="example">This paragraph is blue and center-aligned.</p>
15. </body>
16. </html>
```

Output:



3.8 Box Model (Introduction, Border properties, Padding Properties, Margin properties)

The CSS Box Model

In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:

Explanation of the different parts:

- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content
- **Margin** - Clears an area outside the border. The margin is transparent

PROGRAM:

```
<html>

<head>

<style>

div {

  background-color: lightgrey;

  width: 300px;

  border: 15px solid green;

  padding: 50px;

  margin: 20px;

}

</style>

</head>

<body>
```

```
<h2>Demonstrating the Box Model</h2>

<p>The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.</p>

<div>This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border. </div>

</body>

</html>
```

Output:

Demonstrating the Box Model

The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.

This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border.

Exercise 4

4.1 The Basic of JavaScript: variables and Objects,

In HTML, JavaScript code is inserted between <script> and </script> tags.

JavaScript Variables:

All JavaScript variables must be identified with unique names.

These unique names are called identifiers.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

The general rules for constructing names for variables (unique identifiers) are:

- Names can contain letters, digits, underscores, and dollar signs.
- Names must begin with a letter
- Names can also begin with $ and _ (but we will not use it in this tutorial)
- Names are case sensitive (y and Y are different variables)
- Reserved words (like JavaScript keywords) cannot be used as names

Ways to Declare a JavaScript Variable:

- Using var
- Using let
- Using const

PROGRAM:

```
<html>

<body>

<h1>javascript variables</h1>

<p>x,y,z are variables</p>

<p id="demo"></p>

<p id="demo1"></p>

<p id="demo2"></p>

<p id="demo3"></p>
```

```
<script>
const PI=3.1415926;
{
const PI=3;
document.getElementById("demo").innerHTML=PI;
}
let x=2;
var y=5;
document.getElementById("demo1").innerHTML=PI;
document.getElementById("demo2").innerHTML=x;
document.getElementById("demo3").innerHTML=y;
</script>
</body>
</html>
```

Output:

javascript variables

x,y,z are variables

3

3.1415926

2

5

JavaScript Objects

In real life, a car is an object.

A car has properties like weight and color, and methods like start and stop:

All cars have the same properties, but the property values differ from car to car.

All cars have the same methods, but the methods are performed at different times.

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Objects</h2>
<p id="demo"></p>
<script>
// Create an object:
const person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};
// Display some data from the object:
document.getElementById("demo").innerHTML =
person.firstName + " is " + person.age + " years old.";
</script>
</body>
```

```
</html>
```

Output:

JavaScript Objects

John is 50 years old.

PROGRAM:

```
<html>
<body>
<h2>objects</h2>
<p id="demo"></p>
<script>
const person={fname:"john",lname:"doe",age:50,
fullname: function(){return this.fname+" "+this.lname;}};
document.getElementById("demo").innerHTML=person.fullname();
</script>
</body>
</html>
```

OUTPUT:

objects

john doe

4.2 Primitives Operations and Expressions

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

PROGRAM:

```
<html>
<body>
<p>operators</p>
<p id="demo"></p>
<p id="demo1"></p>
<p id="demo2"></p>
<p id="demo3"></p>
<p id="demo4"></p>
<script>
let a=20;
let b=4;
let c=5.5;
let d=a+b;
let e=a|b;
let f=a-b+c;
let g=a&b;
let h=(100+50)*a/(2*c);
document.getElementById("demo").innerHTML=d;
document.getElementById("demo1").innerHTML=e;
document.getElementById("demo2").innerHTML=f;
document.getElementById("demo3").innerHTML=g;
document.getElementById("demo4").innerHTML=h;
</script>
```

```
</body>

</html>
```

Output:

operators

24

20

21.5

4

272.72727272727275

4.3 Screen Output and Keyboard Input

- The JavaScript model for the HTML document is the Document object
- The model for the browser display window is the Window object
    - The Window object has two properties, document and window, which refer to the document and window objects, respectively
- The Document object has a method, write, which dynamically creates content
    - The parameter is a string, often catenated from parts, some of which are variables
      e.g., document.write("Answer: " + result + "
");
- The Window object has three methods for creating dialog boxes, alert, confirm, and prompt
  The default object for JavaScript is the Window object currently being displayed, so calls to these methods need not include an object reference.
    - alert("Hej! \n");
        - Parameter is plain text, not HTML
        - Opens a dialog box which displays the parameter string and an OK button
    - confirm("Do you want to continue?");
        - Opens a dialog box and displays the parameter and two buttons, OK and Cancel
    - prompt("What is your name?", "");
        - Opens a dialog box and displays its string parameter, along with a text box and two buttons, OK and Cancel
        - The second parameter is for a default response if the user presses OK without typing a response in the text box (waits for OK)

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My First Paragraph</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>
```

```
</body>
</html>
```

Output:

My First Web Page

My First Paragraph.

11

PROGRAM:

```
<!DOCTYPE html>

<html>

<body>

<h2>My First Web Page</h2>

<p>My first paragraph.</p>

<script>

window.alert(5 + 6);

</script>

</body>

</html>
```

OUTPUT:

My First Web Page

My first paragraph.

PROGRAM:

```
<!DOCTYPE html>

<html>

<body>
```

```
<h2>My First Web Page</h2>

<p>My first paragraph.</p>

<p>Never call document.write after the document has finished loading.

It will overwrite the whole document.</p>

<script>

document.write(5 + 6);

</script>

</body>

</html>
```

OUTPUT:

My First Web Page

My first paragraph.

Never call document.write after the document has finished loading. It will overwrite the whole document.

4.4 Object Creation and Modification, Arrays, Functions

Javascript Object Creation :

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Objects</h2>
<p id="demo"></p>
<script>
// Create an object:
const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
// Display some data from the object:
document.getElementById("demo").innerHTML =
person.firstName + " is " + person.age + " years old.";
</script>
</body>
</html>
```

OUTPUT:

JavaScript Objects

John is 50 years old.

PROGRAM:

```
<html>
<body>
<h2>objects</h2>
<p id="demo"></p>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
const p=new Object();
p.fname="john";
p.lname="doe";
p.age=19;
p.eyecolor="black";
document.getElementById("demo").innerHTML=p.fname+" is"+ p["age"]+" years old.";
document.getElementById("demo1").innerHTML=p.fname+ "  "+p.lname;
document.getElementById("demo2").innerHTML=p.fname + " has"+ p.eyecolor+" eyes";
</script>
</body>
</html>
```

OUTPUT:

objects

john is19 years old.

john doe

john hasblack eyes

PROGRAM:

```
<html>
<body>
<h2>getting a property value</h2>
<p id="demo"></p>
<script>
const person={fname:"john",lname:"doe",age:50,eyecolor:"blue",language:"english",
get lang()
{
return this.language;
}
}
document.getElementById("demo").innerHTML=person.lang;
</script>
</body>
</html>
```

OUTPUT:

getting a property value

english

PROGRAM:

```
<html>

<body>

<h2>setting property </h2>

<p id="demo"></p>

<script>

const person={fname:"john",lname:"doe",age:50,eyecolor:"blue",language:"no",

set lang(value)

{

this.language=value;

}

}

person.lang="english";

document.getElementById("demo").innerHTML=person.language;

</script>

</body>

</html>
```

OUTPUT:

setting property

english

PROGRAM:

```
<html>
<body>
<h2>javascript prototype</h2>
<p id="demo"></p>
<p id="demo1"></p>
<script>
function person(first,last,age,eye)
{this.fname=first;
this.lname=last;
this.age=age;
this.eyecolor=eye;
}
person.prototype.nationality="english";
const p=new person("john","doe",50,"blue");
document.getElementById("demo1").innerHTML=p.nationality;
</script>
</body>
</html>
```

OUTPUT:

javascript Prototype

English

Javascript Arrays:

Creating an Array

Using an array literal is the easiest way to create a JavaScript Array.

Syntax:

```
const array_name = [item1, item2, ...];
```

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Arrays</h2>
<p id="demo"></p>
<p id="demo1"></p>
<p id="demo2"></p>
<p id="demo3">we can create array with new array()</p>
<script>
const cars = ["Fiat", "Volvo", "BMW"];
document.getElementById("demo").innerHTML = cars;
document.getElementById("demo1").innerHTML = cars[0];
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo2").innerHTML = fruits.length;
var points = new Array(40, 100, 1);
document.getElementById("demo3").innerHTML = points;
</script>
```

</body>

</html>

OUTPUT:

JavaScript Arrays

Fiat,Volvo,BMW

Fiat

4

40,100,1

PROGRAM:

```
<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Array Methods</h2>

<p>The toString() method returns an array as a comma separated string:</p>

<p>The shift() method returns the element that was shifted out.</p>

<p>The unshift() method adds a new element to an array (at the beginning), and "unshifts" older elements:

The unshift() method returns the new array length.</p>

<p id="demo"></p>

<p id="demo1"></p>

<p id="demo2"></p>

<p id="demo3"></p>

<p id="demo4"></p>

<p id="demo5"></p>
```

```
<p id="demo6"></p>

<p id="demo7"></p>

<script>

const fruits = ["Banana", "Orange", "Apple", "Mango"];

document.getElementById("demo").innerHTML = fruits.toString();

document.getElementById("demo1").innerHTML = fruits.join(" * ");

fruits.pop();

document.getElementById("demo3").innerHTML = fruits;

fruits.push("Kiwi");

document.getElementById("demo4").innerHTML = fruits;

document.getElementById("demo5").innerHTML = fruits.shift();

document.getElementById("demo6").innerHTML = fruits.unshift("Lemon");

document.getElementById("demo7").innerHTML = fruits;

</script>

</body>

</html>
```

OUTPUT:

JavaScript Array Methods

The toString() method returns an array as a comma separated string:

The shift() method returns the element that was shifted out.

The unshift() method adds a new element to an array (at the beginning), and "unshifts" older elements: The unshift() method returns the new array length.

Banana,Orange,Apple,Mango

Banana * Orange * Apple * Mango

Banana,Orange,Apple

Banana,Orange,Apple,Kiwi

Banana

4

Lemon,Orange,Apple,Kiwi

PROGRAM:

```
!DOCTYPE html>

<html>

<body>

<h2>JavaScript Array Methods</h2>

<p>The concat() method merges (concatenates) arrays:</p>

<p>The splice() method adds new items to an array.

Example

const fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.splice(2, 0, "Lemon", "Kiwi");

The first parameter (2) defines the position where new elements should be added (spliced in).

The second parameter (0) defines how many elements should be removed.

The rest of the parameters ("Lemon" , "Kiwi") define the new elements to be added.</p>

<p>The slice() method slices out a piece of an array into a new array.</p>

<p id="demo"></p>

<p id="demo1"></p>

<p id="demo2"></p>

<p id="demo3"></p>

<p id="demo4"></p>
```

```
<script>

const array1 = ["Cecilie", "Lone"];

const array2 = ["Emil", "Tobias", "Linus"];

const array3 = ["Robin", "Morgan"];

const myChildren = array1.concat(array2, array3);

document.getElementById("demo").innerHTML = myChildren;

const fruits = ["Banana", "Orange", "Apple", "Mango"];

document.getElementById("demo1").innerHTML = "Original Array:<br> " + fruits;

let removed = fruits.splice(2, 2, "Lemon", "Kiwi");

document.getElementById("demo2").innerHTML = "New Array:<br>" + fruits;

document.getElementById("demo3").innerHTML = "Removed Items:<br> " + removed;

const citrus = fruits.slice(3);

document.getElementById("demo4").innerHTML = fruits + "<br><br>" + citrus;

</script>

</body>

</html>
```

OUTPUT:

JAVASCRIPT ARRAY METHODS

The concat() method merges (concatenates) arrays:

The splice() method adds new items to an array. Example const fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits.splice(2, 0, "Lemon", "Kiwi"); The first parameter (2) defines the position where new elements should be added (spliced in). The second parameter (0) defines how many elements should be removed. The rest of the parameters ("Lemon" , "Kiwi") define the new elements to be added.

The slice() method slices out a piece of an array into a new array.

Cecilie,Lone,Emil,Tobias,Linus,Robin,Morgan

Original Array:
Banana,Orange,Apple,Mango

New Array:
Banana,Orange,Lemon,Kiwi

Removed Items:
Apple,Mango

Banana,Orange,Lemon,Kiwi

Kiwi

Program:

```
<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Array Sort and reverse</h2>

<p id="demo1"></p>

<p id="demo2"></p>

<script>

const fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.sort();

document.getElementById("demo1").innerHTML = fruits;

fruits.reverse();

document.getElementById("demo2").innerHTML = fruits;

</script>

</body>

</html>
```

Output:

JavaScript Array Sort and reverse

Apple,Banana,Mango,Orange

Orange,Mango,Banana,Apple

JAVASRIPT FUNCTIONS :

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript functions are defined with the function keyword.</h2>
<p>This example calls a function which performs a calculation and returns the result:</p>
<p id="demo"></p>
<script>
var x = myFunction(4, 3);
document.getElementById("demo").innerHTML = x;
function myFunction(a, b) {
  return a * b;
}
</script>
</body>
</html>
```

Output:

JavaScript functions are defined with the function keyword.

This example calls a function which performs a calculation and returns the result:

12

Program:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Functions</h2>
<p>call() method takes arguments separately.<br>
The apply() method takes arguments as an array.:</p>
<p id="demo"></p>
<p id="demo1"></p>
<script>
const person = {
  fullName: function(city, country) {
    return this.firstName + " " + this.lastName + "," + city + "," + country;
  }
}
const person1 = {
  firstName:"John",
  lastName: "Doe"
}
```

```
document.getElementById("demo").innerHTML = person.fullName.call(person1, "Oslo",
"Norway");

document.getElementById("demo1").innerHTML = person.fullName.apply(person1, ["Oslo",
"Norway"]);

</script>

</body>

</html>
```

Output:

JavaScript Functions

call() method takes arguments separately.
The apply() method takes arguments as an array.:

John Doe,Oslo,Norway

John Doe,Oslo,Norway

Program:

```
<!DOCTYPE html>

<html>

<body>

<h2>compare function</h2>

<p>By default, the sort() function sorts values Alphabetically<br>.

However, if numbers are sorted as strings, "25" is bigger than "100",<br>

 because "2" is bigger than "1"<br>.

Because of this, the sort() method will produce incorrect result when sorting numbers.<br>

You can fix this by providing a compare function:

function(a, b){return a - b}
```

```
.</p>

<button onclick="myFunction1()">Sort Alphabetically</button>

<button onclick="myFunction2()">Sort Numerically</button>

<p id="demo"></p>

<script>

const points = [40, 100, 1, 5, 25, 10];

document.getElementById("demo").innerHTML = points;

function myFunction1() {

  points.sort();

  document.getElementById("demo").innerHTML = points;

}

function myFunction2() {

  points.sort(function(a, b){return a - b});

  document.getElementById("demo").innerHTML = points;

}

</script>

</body>

</html>
```

Output:

compare function

By default, the sort() function sorts values Alphabetically
. However, if numbers are sorted as strings, "25" is bigger than "100",
because "2" is bigger than "1"
. Because of this, the sort() method will produce incorrect result when sorting

numbers.
You can fix this by providing a compare function: function(a, b){return a - b} .

Sort Alphabetically Sort Numerically

40,100,1,5,25,10

Program:

```
<!DOCTYPE html>

<html>

<body>

<h2>Max and min number</h2>

<p>You can use Math.min.apply to find the lowest number in an array:<br>

You can use Math.max.apply to find the highest number in an array:</p>

<p id="demo"></p>

<p id="demo1"></p>

<script>

const points = [40, 100, 1, 5, 25, 10];

document.getElementById("demo").innerHTML = myArrayMax(points);

document.getElementById("demo1").innerHTML = myArrayMin(points);

function myArrayMin(arr) {

  return Math.min.apply(null, arr);

}

function myArrayMax(arr) {

  return Math.max.apply(null, arr);

}
```

```
</script>

</body>

</html>
```

Output:

Max and min number

You can use Math.min.apply to find the lowest number in an array:
You can use Math.max.apply to find the highest number in an array:

100

1

Program:

```
<html>

<body>

<h2>map and filter functions</h2>

<p>The map() method creates a new array by performing a function on each array element.<br>

The map() method does not change the original array.<br>

The filter() method creates a new array with array elements that passes a test.

</p>

<p id="demo"></p>

<p id="demo1"></p>

<p id="demo2"></p>

<script>

const numbers1 = [45, 4, 9, 16, 25];
```

```
const numbers2 = numbers1.map(myFunction);

document.getElementById("demo").innerHTML = numbers1;

document.getElementById("demo1").innerHTML = numbers2;

function myFunction(value, index, array)

{

  return value * 2;

}

const over18 = numbers1.filter(myFunction1);

document.getElementById("demo2").innerHTML = over18;

function myFunction1(value, index, array) {

  return value > 18;

}

</script>

</body>

</html>
```

Output:

map and filter functions

The map() method creates a new array by performing a function on each array element.
The map() method does not change the original array.
The filter() method creates a new array with array elements that passes a test.

45,4,9,16,25

90,8,18,32,50

45,25

Exercise 5

5.1 Introducing PHP: Creating PHP script

What is PHP?

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

Basic PHP Syntax

- A PHP script can be placed anywhere in the document.
- A PHP script starts with <?php and ends with ?>:

```
<?php
// PHP code goes here
?>
```

- The default file extension for PHP files is ".php".
- A PHP file normally contains HTML tags, and some PHP scripting code.
- Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

OUTPUT:

My first PHP page

5.2 Running PHP script

PHP Installation for Windows Users: Follow the steps to install PHP on the Windows operating system.

- **Step 1:** First, we have to download PHP from it's official website. We have to download the .zip file from the respective section depending upon on our system architecture(x86 or x64).
- **Step 2:** Extract the .zip file to your preferred location. It is recommended to choose the Boot Drive(C Drive) inside a folder named php (ie. C:\php).
- **Step 3**: Now we have to add the folder (C:\php) to the Environment Variable Path so that it becomes accessible from the command line. To do so, we have to right click on My Computer or This PC icon, then Choose Properties from the context menu. Then click the Advanced system settings link, and then click Environment Variables. In the section System Variables, we have to find the PATH environment variable and then select and Edit it. If the PATH environment variable does not exist, we have to click New. In the Edit System Variable (or New System Variable) window, we have to specify the value of the P

- ATH environment variable (C:\php or the location of our extracted php files). After that, we have to click OK and close all remaining windows by clicking OK.

Comments in PHP

A comment in PHP code is a line that is not executed as a part of the program. Its only purpose is to be read by someone who is looking at the code.

Comments can be used to:

- Let others understand your code
- Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code

PHP supports several ways of commenting:

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>

<?php
// This is a single-line comment

# This is also a single-line comment
?>

</body>
</html>
```

5.3 Using variables , constants , Datatypes , Operators.

Variables

In PHP, a variable is declared using a $ sign followed by the variable name. Here, some important points to know about variables:

- As PHP is a loosely typed language, so we do not need to declare the data types of the variables. It automatically analyzes the values and makes conversions to its correct datatype.
- After declaring a variable, it can be reused throughout the code.
- Assignment Operator (=) is used to assign the value to a variable.

Syntax of declaring a variable in PHP is given below:

1. $variablename=value;

Rules for declaring PHP variable:

- A variable must start with a dollar ($) sign, followed by the variable name.
- It can only contain alpha-numeric character and underscore (A-z, 0-9, _).
- A variable name must start with a letter or underscore (_) character.
- A PHP variable name cannot contain spaces.
- One thing to be kept in mind that the variable name cannot start with a number or special symbols.
- PHP variables are case-sensitive, so $name and $NAME both are treated as different variable.

Variable: Declaring string, integer, and float

Let's see the example to store string, integer, and float values in PHP variables.

PROGRAM:

```
1. <?php
2. $str="hello string";
3. $x=200;
```

```
4. $y=44.6;
5. echo "string is: $str <br/>";
6. echo "integer is: $x <br/>";
7. echo "float is: $y <br/>";
8. ?>
```

Output:

string is: hello string

integer is: 200

float is: 44.6

Variable: case sensitive

In PHP, variable names are case sensitive. So variable name "color" is different from Color, COLOR, COLor etc.

PROGRAM:

```
1. <?php
2. $color="red";
3. echo "My car is " . $color . "<br>";
4. echo "My house is " . $COLOR . "<br>";
5. echo "My boat is " . $coLOR . "<br>";
6. ?>
```

Output:

My car is red

Notice: Undefined variable: COLOR in C:\wamp\www\variable.php on line 4

My house is

Notice: Undefined variable: coLOR in C:\wamp\www\variable.php on line 5

My boat is

Variable: Rules

PHP variables must start with letter or underscore only.

PHP variable can't be start with numbers and special symbols

PROGRAM:

```
<?php
$a="hello";//letter (valid)
$_b="hello";//underscore (valid)

echo "$a <br/> $_b";
?>
```

Output:

hello

hello

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<p>In PHP, a variable starts with the $ sign, followed by the name of the variable:</p>
<p>A variable name must start with a letter or the underscore character</p>
<p>The PHP echo statement is often used to output data to the screen.</p>
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
echo $txt;
echo "<br>";
echo $x;
echo "<br>";
echo $y;
?>
</body>
</html>
```

OUTPUT:

In PHP, a variable starts with the $ sign, followed by the name of the variable:

A variable name must start with a letter or the underscore character

The PHP echo statement is often used to output data to the screen.

"; echo $x; echo "

Variable Scope

The scope of a variable is defined as its range in the program under which it can be accessed. In other words, "The scope of a variable is the portion of the program within which it is defined and can be accessed."

PHP has three types of variable scopes:

1. Local variable
2. Global variable
3. Static variable

Local variable

The variables that are declared within a function are called local variables for that function. These local variables have their scope only in that particular function in which they are declared. This means that these variables cannot be accessed outside the function, as they have local scope.

A variable declaration outside the function with the same name is completely different from the variable declared inside the function. Let's understand the local variables with the help of an example

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<p>A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function:</p>
<?php
function myTest() {
  $x = 5; // local scope
  echo "<p>Variable x inside function is: $x</p>";
}
myTest();
// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
</body>
</html>
```

OUTPUT:

A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function:

Variable x inside function is: $x
"; } myTest(); // using x outside the function will generate an error echo "

Variable x outside function is: $x

"; ?>

Global variable

The global variables are the variables that are declared outside the function. These variables can be accessed anywhere in the program. To access the global variable within a function, use the GLOBAL keyword before the variable. However, these variables can be directly accessed or used outside the function without any keyword. Therefore there is no need to use any keyword to access a global variable outside the function.

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<p>PHP has three different variable scopes:
local<br>
global<br>
static</p>
<p>A variable declared outside a function has a GLOBAL SCOPE and can only be accessed
outside a function:</p>
<p>The global keyword is used to access a global variable  within a function.</p>
<?php
$x = 5; // global scope
function myTest() {
global $x;
  // using x inside this function will generate an error
  echo "<p>Variable x inside function is: $x</p>";
}
myTest();
echo "<p>Variable x outside function is: $x</p>";
?>
</body>
</html>
```

OUTPUT:

PHP has three different variable scopes: local
global
static

A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function:

The global keyword is used to access a global variable within a function.

Variable x inside function is: $x
"; } myTest(); echo "

Variable x outside function is: $x

"; ?>

Static variable

It is a feature of PHP to delete the variable, once it completes its execution and memory is freed. Sometimes we need to store a variable even after completion of function execution. Therefore, another important feature of variable scoping is static variable. We use the static keyword before the variable to define a variable, and this variable is called as static variable.

Static variables exist only in a local function, but it does not free its memory after the program execution leaves the scope.

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<?php
function myTest() {
  static $x = 0;
  echo $x;
  $x++;
}
myTest();
echo "<br>";
myTest();
echo "<br>";
myTest();
```

```
?>
</body>
</html>
```

OUTPUT:

```
";myTest();echo                                                    "
"; myTest(); ?>
```

Constants

PHP constants are name or identifier that can't be changed during the execution of the script except for magic constants, which are not really constants. PHP constants can be defined by 2 ways:

Using define() function

Using const keyword

Constants are similar to the variable except once they defined, they can never be undefined or changed. They remain constant across the entire program. PHP constants follow the same PHP variable rules. For example, it can be started with a letter or underscore only.

Conventionally, PHP constants should be defined in uppercase letters.

Note: Unlike variables, constants are automatically global throughout the script.

constant: define()

Use the define() function to create a constant. It defines constant at run time. Let's see the syntax of define() function in PHP

define(name, value, **case**-insensitive)

1. name**:** It specifies the constant name.
2. value**:** It specifies the constant value.
3. case-insensitive**:** Specifies whether a constant is case-insensitive. Default value is false. It means it is case sensitive by default.

Let's see the example to define PHP constant using define().

PROGRAM:

```
<?php
```

```
define("MESSAGE","Hello JavaTpoint PHP");

echo MESSAGE;

?>
```

Output:

```
Hello JavaTpoint PHP
```

Create a constant with **case-insensitive** name:

PROGRAM:

```
<?php

define("MESSAGE","Hello JavaTpoint PHP",true);//not case sensitive

echo MESSAGE, "</br>";

echo message;

?>
```

Output:

```
Hello JavaTpoint PHP

Hello JavaTpoint PHP
```

Constant: const keyword

PHP introduced a keyword const to create a constant. The const keyword defines constants at compile time. It is a language construct, not a function. The constant defined using const keyword are case-sensitive.

PROGRAM:

```
<?php

const MESSAGE="Hello const by JavaTpoint PHP";

echo MESSAGE;

?>
```

Output:

Hello const by JavaTpoint PHP

Constant() function

There is another way to print the value of constants using constant() function instead of using the echo statement.

Syntax

The syntax for the following constant function:

1. constant (name)

PROGRAM:

```
1. <?php
2.     define("MSG", "JavaTpoint");
3.     echo MSG, "</br>";
4.     echo constant("MSG");
5.     //both are similar
6. ?>
```

Output:

JavaTpoint

JavaTpoint

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<p>
To create a constant, use the define() function.<br>
Syntax:
define(name, value, case-insensitive)<br>
Constants are automatically global and can be used across the entire script.<br>
A constant is an identifier (name) for a simple value. The value cannot be changed during the script.<br>
A valid constant name starts with a letter or underscore (no $ sign before the constant name).
```

```
</p>
<?php
// case-sensitive constant name
define("GREETING", "Welcome!");
echo "<br>";
echo GREETING;
echo "<br>";
define("cars", [
  "Audi",
  "BMW",
  "Toyota"
]);
echo cars[0];
echo "<br>";
//constant is global but can be used inside the function
define("GREETING1", "Welcome!");
function myTest() {
  echo GREETING1;
}
myTest();
?>
</body>
</html>
```

OUTPUT:

To create a constant, use the define() function.
Syntax: define(name, value, case-insensitive)
Constants are automatically global and can be used across the entire script.
A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
A valid constant name starts with a letter or underscore (no $ sign before the constant name).

"; echo GREETING; echo "
"; define("cars", ["Audi", "BMW", "Toyota"]); echo cars[0]; echo "
"; //constant is global but can be used inside the function define("GREETING1", "Welcome!");
function myTest() { echo GREETING1; } myTest(); ?>

Data Types

PHP data types are used to hold different types of data or values. PHP supports 8 primitive data types that can be categorized further in 3 types:

1. Scalar Types (predefined)
2. Compound Types (user-defined)

3. Special Types

Data Types: Scalar Types

It holds only single value. There are 4 scalar data types in PHP.

- boolean
- integer
- float
- string

Data Types: Compound Types

It can hold multiple values. There are 2 compound data types in PHP.

1. Array
2. object

Data Types: Special Types

There are 2 special data types in PHP

- resource
- NULL

Boolean

Booleans are the simplest data type works like switch. It holds only two values: TRUE **(1)** or FALSE **(0)**. It is often used with conditional statements. If the condition is correct, it returns TRUE otherwise FALSE.

PROGRAM:

```
1. <?php
2.     if (TRUE)
3.         echo "This condition is TRUE.";
4.     if (FALSE)
5.         echo "This condition is FALSE.";
6. ?>
```

Output:

This condition is TRUE.

Integer

Integer means numeric data with a negative or positive sign. It holds only whole numbers, i.e., numbers without fractional part or decimal points.

Rules for integer:

- An integer can be either positive or negative.
- An integer must not contain decimal point.
- Integer can be decimal (base 10), octal (base 8), or hexadecimal (base 16).
- The range of an integer must be lie between 2,147,483,648 and 2,147,483,647 i.e., -2^31 to 2^31.

PROGRAM:

```
<?php
    $dec1 = 34;
    $oct1 = 0243;
    $hexa1 = 0x45;
    echo "Decimal number: " .$dec1. "</br>";
    echo "Octal number: " .$oct1. "</br>";
    echo "HexaDecimal number: " .$hexa1. "</br>";
?>
```

Output:

Decimal number: 34

Octal number: 163

HexaDecimal number: 69

Float

A floating-point number is a number with a decimal point. Unlike integer, it can hold numbers with a fractional or decimal point, including a negative or positive sign.

PROGRAM:

```php
<?php
    $n1 = 19.34;
    $n2 = 54.472;
    $sum = $n1 + $n2;
    echo "Addition of floating numbers: " .$sum;
?>
```

Output:

Addition of floating numbers: 73.812

String

A string is a non-numeric data type. It holds letters or any alphabets, numbers, and even special characters.

String values must be enclosed either within single quotes or in double quotes. But both are treated differently. To clarify this, see the example below:

PROGRAM:

```php
<?php
    $company = "Javatpoint";
    //both single and double quote statements will treat different
    echo "Hello $company";
    echo "</br>";
    echo 'Hello $company';
?>
```

Output:

Hello Javatpoint

Hello $company

Array

An array is a compound data type. It can store multiple values of same data type in a single variable.

1. **PROGRAMS**

```
<?php
    $bikes = array ("Royal Enfield", "Yamaha", "KTM");
    var_dump($bikes);   //the var_dump() function returns the datatype and values
    echo "</br>";
    echo "Array Element1: $bikes[0] </br>";
    echo "Array Element2: $bikes[1] </br>";
    echo "Array Element3: $bikes[2] </br>";
?>
```

Output:

array(3) { [0]=> string(13) "Royal Enfield" [1]=> string(6) "Yamaha" [2]=> string(3) "KTM" }

Array Element1: Royal Enfield

Array Element2: Yamaha

Array Element3: KTM

OBJECT

Objects are the instances of user-defined classes that can store both values and functions. They must be explicitly declared.

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<?php
class Car {
  public $color;
  public $model;
  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }
  public function message() {
    return "My car is a " . $this->color . " " . $this->model . "!";
  }
}
```

```
$myCar = new Car("black", "Volvo");
echo $myCar -> message();
echo "<br>";
$myCar = new Car("red", "Toyota");
echo $myCar -> message();
?>
</body>
</html>
```

OUTPUT:

color = $color; $this->model = $model; } public function message() { return "My car is a " . $this->color . " " . $this->model . "!"; } } $myCar = new Car("black", "Volvo"); echo $myCar -> message(); echo "
"; $myCar = new Car("red", "Toyota"); echo $myCar -> message(); ?>

Operators:

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<?php
$x = 10;
$y = 3;
echo $x ** $y;
echo "<br>";
//spaceship operator
$x = 5;
$y = 10;
echo ($x <=> $y); // returns -1 because $x is less than $y
echo "<br>";
$x = 10;
$y = 10;
echo ($x <=> $y); // returns 0 because values are equal
echo "<br>";
$x = 15;
$y = 10;
echo ($x <=> $y); // returns +1 because $x is greater than $y
echo "<br>";
//concatenation operator
$txt1 = "Hello";
$txt2 = " world!";
$txt1 .= $txt2;
echo $txt1;
echo "<br>";
```

```
//Identity operator
$x = array("a" => "red", "b" => "green");
$y = array("c" => "blue", "d" => "yellow");
$z = array( "b" => "green","a" => "red",);
$k = array("a" => "red", "b" => "green");
var_dump($x === $z);
echo "<br>";
var_dump($x == $z);
echo"<br>";
var_dump($x === $k);
echo"<br>";
//Null coalescing operator
  // variable $user is the value of $_GET['user']
  // and 'anonymous' if it does not exist
  echo $user = $_GET["user"] ?? "anonymous";
  echo("<br>");
  // variable $color is "red" if $color does not exist or is null
  echo $color = $color ?? "red";
?>
</body>
</html>
```

OUTPUT:

```
"; //spaceship operator $x = 5; $y = 10; echo ($x <=> $y); // returns -1 because $x is less than $y
echo "
"; $x = 10; $y = 10; echo ($x <=> $y); // returns 0 because values are equal echo "
"; $x = 15; $y = 10; echo ($x <=> $y); // returns +1 because $x is greater than $y echo "
"; //concatenation operator $txt1 = "Hello"; $txt2 = " world!"; $txt1 .= $txt2; echo $txt1; echo "
"; //Identity operator $x = array("a" => "red", "b" => "green"); $y = array("c" => "blue", "d" =>
"yellow"); $z = array( "b" => "green","a" => "red",); $k = array("a" => "red", "b" => "green");
var_dump($x === $z); echo "
"; var_dump($x == $z); echo"
"; var_dump($x === $k); echo"
"; //Null coalescing operator // variable $user is the value of $_GET['user'] // and 'anonymous' if
it does not exist echo $user = $_GET["user"] ?? "anonymous"; echo("
"); // variable $color is "red" if $color does not exist or is null echo $color = $color ?? "red"; ?>
```

5.4 Conditional statements, Control statements, Arrays ,functions

IF-ELSE STATEMENT

PHP if-else statement is executed whether condition is true or false.

If-else statement is slightly different from if statement. It executes one block of code if the specified condition is true and another block of code if the condition is false.

PROBLEM:

```
<!DOCTYPE html>
<html>
<body>
<?php
$x = 50;
$y=20;
if ($x < $y)
{
  echo "y is greater";
}
else
{
  echo "x is greater";
}
?>
</body>
</html>
```

IF-ELSE-IF STATEMENT

The PHP if-else-if is a special statement used to combine multiple if?.else statements. So, we can check multiple conditions using this statement.

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<?php
$x =20;
$y=20;
```

```
if ($x < $y)
{
  echo "y is greater";
}
else if ($x > $y)
{
  echo "x is greater";
}
else
{
echo "x and y are equal";
}
?>
</body>
</html>
```

OUTPUT:

```
y) { echo "x is greater"; } else { echo "x and y are equal"; } ?>
```

Switch

```
<!DOCTYPE html>
<html>
<body>

<?php
$favcolor = "red";

switch ($favcolor) {
  case "red":
    echo "Your favorite color is red!";
    break;
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
    echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

```
</body>
</html>
```

Array

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<p>sort() - sort arrays in ascending order<br>
rsort() - sort arrays in descending order
</p>
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);

$clength = count($cars);
for($x = 0; $x < $clength; $x++) {
  echo $cars[$x];
  echo "<br>";
}
?>

</body>
</html>
```

OUTPUT:

sort() - sort arrays in ascending order
rsort() - sort arrays in descending order

```
"; } ?>
```

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<p>
asort() - sort associative arrays in ascending order, according to the value<br>
ksort() - sort associative arrays in ascending order, according to the key<br>
arsort() - sort associative arrays in descending order, according to the value<br>
krsort() - sort associative arrays in descending order, according to the key</p>
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

```
echo "Peter is " . $age['Peter'] . " years old.";
 echo "<br>";
echo " sort in ascending order according to value <br>";
asort($age);
foreach($age as $x => $x_value) {
  echo "Key=" . $x . ", Value=" . $x_value;
  echo "<br>";
}
echo " sort in decending order according to key <br>";
krsort($age);
foreach($age as $x => $x_value) {
  echo "Key=" . $x . ", Value=" . $x_value;
  echo "<br>";
}
?>
</body>
</html>
```

OUTPUT:

asort() - sort associative arrays in ascending order, according to the value
ksort() - sort associative arrays in ascending order, according to the key
arsort() - sort associative arrays in descending order, according to the value
krsort() - sort associative arrays in descending order, according to the key

"35", "Ben"=>"37", "Joe"=>"43"); echo "Peter is " . $age['Peter'] . " years old."; echo "
"; echo " sort in ascending order according to value
"; asort($age); foreach($age as $x => $x_value) { echo "Key=" . $x . ", Value=" . $x_value; echo "
"; } echo " sort in decending order according to key
"; krsort($age); foreach($age as $x => $x_value) { echo "Key=" . $x . ", Value=" . $x_value; echo "
"; } ?>

While Loop

PHP while loop can be used to traverse set of code like for loop. The while loop executes a block of code repeatedly until the condition is FALSE. Once the condition gets FALSE, it exits from the body of loop.

It should be used if the number of iterations is not known.

The while loop is also called an Entry control loop because the condition is checked before entering the loop body. This means that first the condition is checked. If the condition is true, the block of code will be executed.

Syntax

1. **while**(condition){
2. //code to be executed
3. }

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>
<?php
echo "While loop <br>";
$x = 1;
while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
  }
echo "do While loop <br>";
$x = 6;
do {
  echo "The number is: $x <br>";
  $x++;
} while ($x <= 5);
echo "for loop <br>";
for ($x = 0; $x <= 6; $x++) {
  echo "The number is: $x <br>";
  }
echo "for each loop <br>";
 $colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
  echo "$value <br>";
}
echo "break <br>";
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    break;
  }
  echo "The number is: $x <br>";
}
echo "continue <br>";
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    continue;
  }
  echo "The number is: $x <br>";
```

```
}
?>
</body>
</html>
```

OUTPUT:

```
"; $x = 1; while($x <= 5) { echo "The number is: $x
"; $x++; } echo "do While loop
"; $x = 6; do { echo "The number is: $x
"; $x++; } while ($x <= 5); echo "for loop
"; for ($x = 0; $x <= 6; $x++) { echo "The number is: $x
"; } echo "for each loop
"; $colors = array("red", "green", "blue", "yellow"); foreach ($colors as $value) { echo "$value
"; } echo "break
"; for ($x = 0; $x < 10; $x++) { if ($x == 4) { break; } echo "The number is: $x
"; } echo "continue
"; for ($x = 0; $x < 10; $x++) { if ($x == 4) { continue; } echo "The number is: $x
"; } ?>
```

Functions

PHP function is a piece of code that can be reused many times. It can take input as argument list and return value. There are thousands of built-in functions in PHP.

PROGRAM:

```
<!DOCTYPE html>
<html>
<body>

<?php

//return value
function sum(int $x, int $y)
{
  $z = $x + $y;
  return $z;
}
echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
echo "<br>";
```

```
//To declare a type for the function return, add a colon ( : )

function addNumbers2(float $a, float $b) : float {
  return $a + $b;
}
echo addNumbers2(1.2, 5.2);
echo "<br>";
//reference
function add_five(&$value)
{
  $value += 5;
}

$num = 2;
add_five($num);
echo $num;
?>

</body>
</html>
```

OUTPUT:

```
"; echo "7 + 13 = " . sum(7, 13) . "
"; echo "2 + 4 = " . sum(2, 4); echo "
"; //To declare a type for the function return, add a colon ( : ) function addNumbers2(float $a,
float $b) : float { return $a + $b; } echo addNumbers2(1.2, 5.2); echo "
"; //reference function add_five(&$value) { $value += 5; } $num = 2; add_five($num); echo
$num; ?>
```

EXAMPLE CODE:

```
<?php declare(strict_types=1); // strict requirement

function addNumbers(int $a, int $b) {
  return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is enabled and "5 days" is not an integer, an error will be thrown
?>
```

5.5 Working with forms and Databases such as MySQL.

With PHP, you can connect to and manipulate databases.

MySQL is the most popular database system used with PHP.

What is MySQL?

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation
- MySQL is named after co-founder Monty Widenius's daughter:

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

Databases are useful for storing information categorically. A company may have a database with the following tables:

- Employees
- Products
- Customers
- Orders

PHP + MySQL Database System

- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

Database Queries

A query is a question or a request.

We can query a database for specific information and have a recordset returned.

Look at the following query (using standard SQL):

SELECT LastName FROM Employees

The query above selects all the data in the "LastName" column from the "Employees" table.

To learn more about SQL, please visit our SQL tutorial.

Download MySQL Database

If you don't have a PHP server with a MySQL Database, you can download it for free here: http://www.mysql.com

5.6 Develop PHP MySQL CRUD Application

P insertform.html:

PROGRAM:

P insertform.html:

```
<html>
    <body>
      <form action="insert.php" method="post">
        ID <input type="text" name="id"/><br>
        Name <input type="text" name="name"/><br>
        Marks <input type="text" name="marks"/><br>
        <input type="submit" value="insert"/><br>
      </form>
    </body>
</html>
```

Insert.php:

```
<?php
    $server="localhost";
    $user="root";
    $pass="";
    $dbname="srkr";
    $con =mysqli_connect($server,$user,$pass,$dbname);
    if(!$con)
    {
      die("Error : :  ".mysqli_connect_error());
    }
    $sql = "INSERT INTO student VALUES
('{$_POST['id']}','{$_POST['name']}','{$_POST['marks']}')";
    $res = mysqli_query($con,$sql);
    if($res)
    {
      $sql2="select * from student";
      $sres=mysqli_query($con,$sql2);
      echo "<table border=1>
            <tr>
             <th>ID</th>
             <th>Name</th>
             <th>Marks</th>
            </tr>";
      while($row= mysqli_fetch_array($sres))
      {
        echo "<tr>";
```

```
      echo "<td>".$row[0]."</td>";
      echo "<td>".$row[1]."</td>";
      echo "<td>".$row[2]."</td>";
      echo "</tr>";
   }
   echo "</table>";
}
else
{
    echo "Issue with query";
}
mysqli_close($con);
```

```
?>
```

Update.html:

```
<html>
    <body>
      <form action="update.php" method="post">
        ID <input type="text" name="id"/>
        <input type="submit" value="update"/><br>
        Name <input type="text" name="name"/><br>
      </form>
    </body>
</html>
```

Update.php:

```
<?php
    $server="localhost";
    $user="root";
    $pass="";
    $dbname="srkr";
    $con =mysqli_connect($server,$user,$pass,$dbname);
    if(!$con)
    {
      die("Error : :  ".mysqli_connect_error());
    }
    $sql = "UPDATE student SET name='{$_POST['name']}' WHERE
  id='{$_POST['id']}'";
    $res = mysqli_query($con,$sql);
    if($res)
    {
      $sql2="select * from student";
      $sres=mysqli_query($con,$sql2);
      echo "<table border=1>
          <tr>
          <th>ID</th>
          <th>Name</th>
          <th>Marks</th>
          </tr>";
      while($row= mysqli_fetch_array($sres))

      {
        echo "<tr>";
        echo "<td>".$row[0]."</td>";
        echo "<td>".$row[1]."</td>";
```

```
                    echo "<td>".$row[2]."</td>";
                    echo "</tr>";
                  }
                  echo "</table>";
                }
                else
                {
                   echo "Issue with query";
                }
                mysqli_close($con);

            ?>
```

Delete.html:

```
<html>
   <body>
      <form action="delete.php" method="post">
         ID <input type="text" name="id"/><br>
<input type="submit" value="delete"/><br>
      </form>
   </body>
</html>
```

Delete.php:

```
<?php
   $server="localhost";
   $user="root";
   $pass="";
   $dbname="srkr";
   $con =mysqli_connect($server,$user,$pass,$dbname);
   if(!$con)
   {
      die("Error : :  ".mysqli_connect_error());
   }
```

```
    $sql = "DELETE FROM student WHERE id='{$_POST['id']}'";
    $res = mysqli_query($con,$sql);

    if($res)
    {
      $sql2="select * from student";
      $sres=mysqli_query($con,$sql2);
      echo "<table border=1>
            <tr>
             <th>ID</th>
             <th>Name</th>
             <th>Marks</th>
            </tr>";
      while($row= mysqli_fetch_array($sres))
      {
         echo "<tr>";
         echo "<td>".$row[0]."</td>";
         echo "<td>".$row[1]."</td>";
         echo "<td>".$row[2]."</td>";
         echo "</tr>";
      }
      echo "</table>";
    }
else
{
  echo "Issue with query";
    }
    mysqli_close($con);
```

?>