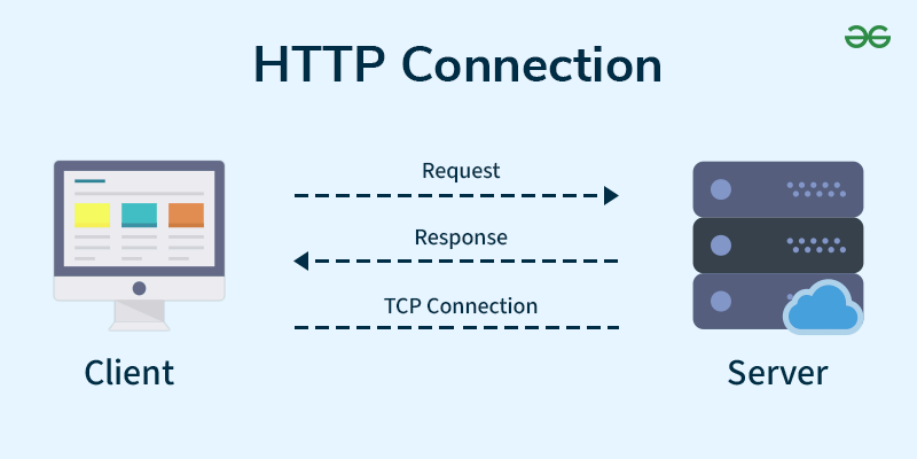
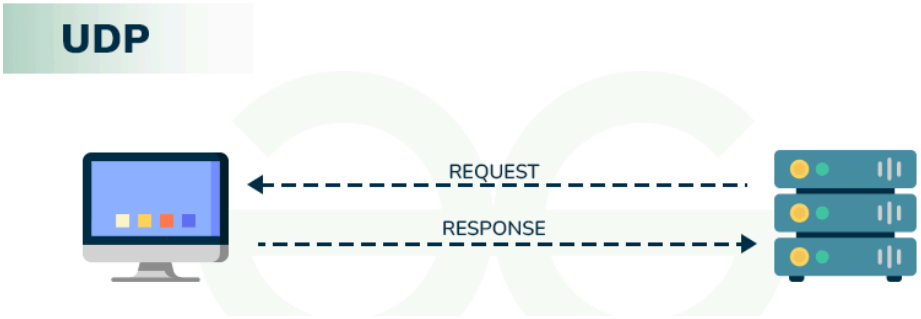
CMR INSTITUTE OF TECHNOLOGY	USN	

Internal Assessment Test –I, June 2025				
Sub :	**Web Application Development**	**Code:**	**MMC205**	
Answer Key		**Marks**	**OBE**	
			CO	RBT
1	Briefly Explain the Following: i) HTTP ii) UDP iii) TCP iv) FTP v) SMTP 1. **Hypertext Transfer Protocol** HTTP protocol is used to transfer hypertexts over the internet and it is defined by the www(world wide web) for information transfer. This protocol defines how the information needs to be formatted and transmitted. And, it also defines the various actions the web browsers should take in response to the calls made to access a particular web page. Whenever a user opens their web browser, the user will indirectly use HTTP as this is the protocol that is being used to share text, images, and other multimedia files on the World Wide Web.  **2. User Datagram Protocol (UDP)** UDP is a communication protocol used for time-sensitive transmissions such as video playback. **How It Works**: UDP is a connectionless protocol that sends messages, called datagrams, without establishing a connection between the sender and receiver. It is faster but less reliable than TCP.	[10]	CO1	L1

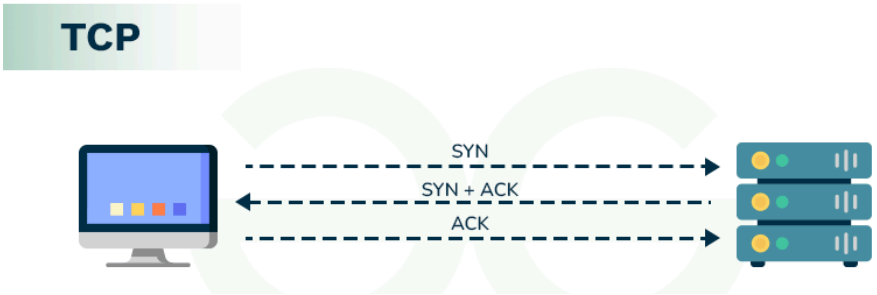
Security Considerations: UDP does not provide mechanisms for ensuring message integrity or order; security must be handled by the application layer.

Use Cases: Streaming media, online gaming..



3. **TCP/IP(Transmission Control Protocol/ Internet Protocol)**

In TCP/IP, the IP protocol ensures that each computer that is connected to the Internet is having a specific serial number called the IP address. TCP specifies how data is exchanged over the internet and how it should be broken into IP packets. It also makes sure that the packets have information about the source of the message data, the destination of the message data, the sequence in which the message data should be re-assembled, and checks if the message has been sent correctly to the specific destination. The TCP is also known as a connection-oriented protocol.



4.**What is File Transfer Protocol?**

File Transfer Protocol (FTP) is a standard network protocol used for transferring files between a client and a server over a TCP/IP network. Developed in the early 1970s, FTP operates at the application layer of the OSI model and facilitates the uploading, downloading, and management of files on remote systems.

🔧 How FTP Works

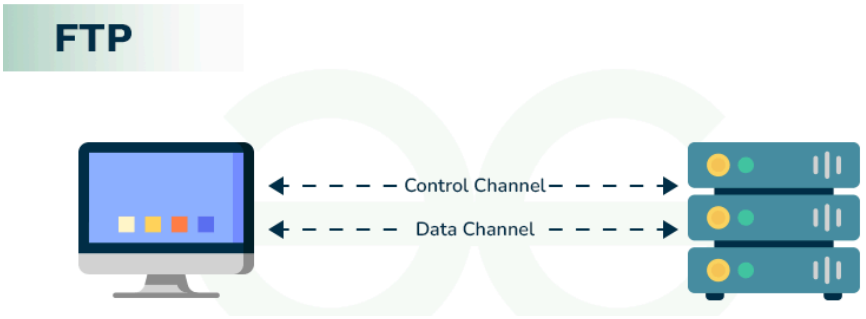
FTP employs a client-server architecture and utilizes two separate channels:

- **Control Channel (Port 21):** Handles commands and responses between the client and server.

- **Data Channel (Port 20):** Transfers the actual files.

There are two modes of FTP operation:

- **Active Mode:** The client opens a port and listens while the server actively connects to it.
- **Passive Mode:** The server opens a port and waits for the client to connect, which is useful when the client is behind a firewall.



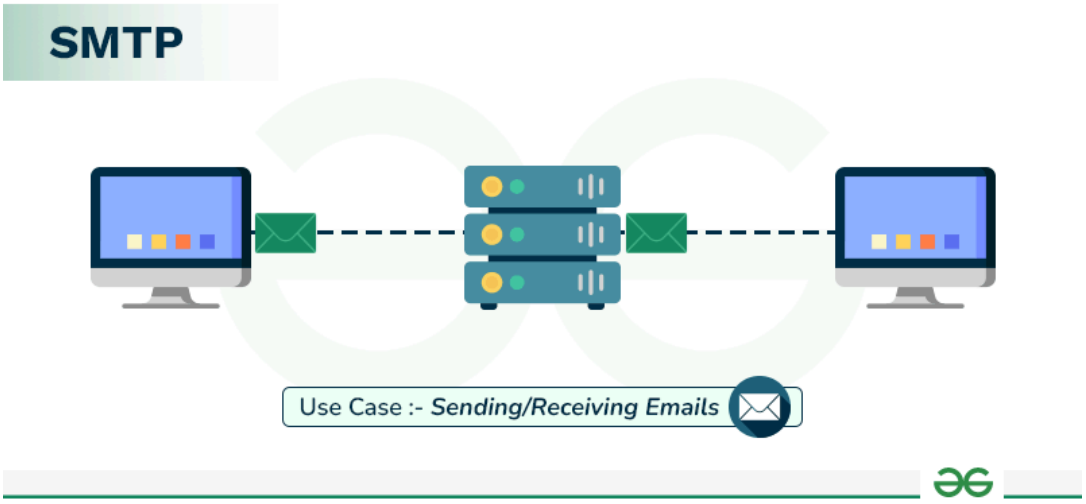
File Transfer Protocol

The **File Transfer Protocol (FTP)** is widely used in the application layer of networking. It works at the application layer, ensuring that files are sent and received securely.

5.**Simple Mail Transfer Protocol (SMTP)**

Simple Mail Transfer Protocol (SMTP) is an application layer protocol used for exchanging email messages between servers. It is essential in the email communication process and operates at the application layer of the TCP/IP stack.

To send an email, the client opens a TCP connection to the SMTP server. The server, which is always listening on port 25, initiates the connection as soon as it detects a client. Once the TCP connection is established, the client sends the email across the connection.



SMTP

Types of SMTP Protocol

The **SMTP model** supports two types of email delivery methods: **end-to-end** and **store-and-forward**.

- **End-to-end** delivery is used between organizations. In this method, the email is sent directly from the sender's SMTP client to the recipient's SMTP server without passing through intermediate servers.
- **Store-and-forward** is used within organizations that have TCP/IP and SMTP-based networks. In this method, the email may pass through several intermediate servers (Message Transfer Agents, or MTAs) before reaching the recipient.

With **end-to-end** delivery, the SMTP client waits until the email is successfully copied to the recipient's SMTP server before sending it. This is different from the **store-and-forward** method, where the email might stop at multiple intermediate servers before reaching its destination. In store-and-forward systems, the sender is notified as soon as the email reaches the first server, not the final destination.

2	Create a Registration form to accept name,Password, Email, Phone number, gender, date of birth, qualification,address and provide Reset and Submit buttons. <!DOCTYPE html> <html> <head> <title>Registration Form</title> </head> <body> Registration Form <form> <label for="name">Name:</label> <input type="text" id="name" name="name" required> 	[10]	CO2	L3

```
    <label for="password">Password:</label><br>
    <input type="password" id="password" name="password" required><br><br>

    <label for="email">Email:</label><br>
    <input type="email" id="email" name="email" required><br><br>

    <label for="phone">Phone Number:</label><br>
    <input type="tel" id="phone" name="phone" pattern="[0-9]{10}" required><br><br>

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

    <label for="dob">Date of Birth:</label><br>
    <input type="date" id="dob" name="dob" required><br><br>

    <label for="qualification">Qualification:</label><br>
    <select id="qualification" name="qualification" required>
        <option value="">--Select--</option>
        <option value="High School">High School</option>
        <option value="Diploma">Diploma</option>
        <option value="Undergraduate">Undergraduate</option>
        <option value="Postgraduate">Postgraduate</option>
        <option value="PhD">PhD</option>
    </select><br><br>

    <label for="address">Address:</label><br>
    <textarea id="address" name="address" rows="4" cols="40"
required></textarea><br><br>

    <input type="submit" value="Submit">
    <input type="reset" value="Reset">
</form>

</body>
</html>
```

3	Explain Box model in detail with an Example. • **BOX Model** The **CSS Box Model** is a fundamental concept in web design that describes how elements are structured and spaced on a webpage. Every element in CSS is treated as a rectangular box, and the box model determines its size, spacing, and positioning. **Structure of the CSS Box Model** The box model consists of the following components (from inside out):	[10]	CO2	L2

1. **Content**

The actual content of the box, such as text, images, or other elements.

Size controlled by: width, height.

div { width: 200px; height: 100px; }

2. **Padding**

The space between the content and the border.

Size controlled by: padding property.

Padding increases the overall size of the box.

Property:

padding-top
padding-right
padding-bottom
padding-left

padding: 10px;

Example:-

```
div {
        padding: 10px; /* Adds 10px padding on all sides */
        padding-left: 20px; /* Adds 20px padding to the left only */
    }
```

3. **Border**

A line surrounding the padding and content.

Size controlled by: border-width, border-style, border-color.

Property:-

border-width
border-style (e.g., solid, dashed, dotted, none)
border-color

Example:-

```
border: 2px solid black;
div
 {
        border: 2px solid black; /* 2px solid border */
        border-radius: 10px; /* Rounded corners */
```

}

4. **Margin**

The space between the element and its neighboring elements.

Size controlled by: margin property.

Property:-
margin-top

margin-right

margin-bottom

margin-left
margin: 20px;

Example:

div {

margin: 15px; /* Adds 15px margin on all sides */

margin-top: 20px; /* Adds 20px margin to the top only */

}

Box Sizing

- The box-sizing property defines whether the width and height include the padding and border or not.

div {

box-sizing: border-box; /* Includes padding and border in width/height */

}

Example:-

div {

width: 200px;

height: 100px;

padding: 10px;

border: 5px solid black;

	margin: 20px; box-sizing: border-box; } 			
4	List out the various bootstrap List Groups classes for different styles of List. Explain their use with code snippets. **Basic List Groups** The most basic list group is an unordered list with list items: • First item • Second item • Third item To create a basic list group, use an element with class .list-group, and - elements with class .list-group-item: <ul class="list-group"> <li class="list-group-item">First item <li class="list-group-item">Second item <li class="list-group-item">Third item **List Group With Badges** You can also add badges to a list group. The badges will automatically be positioned on the right: • New3 • Deleted5 • Warnings3 To create a badge, create a element with class .badge inside the list item: <ul class="list-group"> <li class="list-group-item">New 12	[10]	CO3	L2

```
  <li class="list-group-item">Deleted <span class="badge">5</span></li>

  <li class="list-group-item">Warnings <span class="badge">3</span></li>

</ul>
```

List Group With Linked Items

The items in a list group can also be hyperlinks. This will add a grey background color on hover:

First item

Second item

Third item

To create a list group with linked items, use instead of <ul> and <a> instead of <li>:

```
<div class="list-group">

  <a href="#" class="list-group-item">First item</a>

  <a href="#" class="list-group-item">Second item</a>

  <a href="#" class="list-group-item">Third item</a>

</div>
```

Active State

First item

Second item

Third item

Use the .active class to highlight the current item:

```
<div class="list-group">

  <a href="#" class="list-group-item active">First item</a>

  <a href="#" class="list-group-item">Second item</a>

  <a href="#" class="list-group-item">Third item</a>
```

Disabled Item

The following list group has a disabled item:

First item

Second item

Third item

To disable an item, add the .disabled class

```
<div class="list-group">
  <a href="#" class="list-group-item disabled">First item</a>
  <a href="#" class="list-group-item">Second item</a>
  <a href="#" class="list-group-item">Third item</a>
</div>
```

Contextual Classes

Contextual classes can be used to color list items:

- First item
- Second item
- Third item
- Fourth item

The classes for coloring list-items are: .list-group-item-success, list-group-item-info, list-group-item-warning, and .list-group-item-danger:

```
<ul class="list-group">
  <li class="list-group-item list-group-item-success">First item</li>
  <li class="list-group-item list-group-item-info">Second item</li>
  <li class="list-group-item list-group-item-warning">Third item</li>
```

	<li class="list-group-item list-group-item-danger">Fourth item 			

5	Discuss various selectors of CSS. Explain its usage with an example.
	1)CSS Selector
	CSS selectors are used *to select the content you want to style*. Selectors are part of the CSS rule set. CSS selectors select HTML elements according to its id, class, type, attribute etc.
	
	There are several different types of selectors in CSS.
	
	1. CSS Element Selector
	2. CSS Id Selector
	3. CSS Class Selector
	4. CSS Universal Selector
	5. CSS Group Selector
	
	1) CSS Element Selector
	The element selector selects the HTML element by name.
	
	Example:-
	
	<!DOCTYPE html>
	<html>
	<head>
	<style>
	p{
	text-align: center;
	color: blue;
	}
	</style>
	</head>
	<body>
	 This style will be applied on every paragraph.
	 Me too!
	 And me!
	<h1>hello
	</body>
	</html>
	Output:-
	This style will be applied on every paragraph.
	
	Me too!
		[10]	CO2	L1

And me!

2) CSS Id Selector

The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element.

It is written with the hash character (#), followed by the id of the element.

Let?s take an example with the id "para1".

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
   text-align: center;
   color: blue;
}
</style>
</head>
<body>
<p id="para1">Hello Javatpoint.com</p>
<p>This paragraph will not be affected.</p>
<h1 id=”para1”>cmrit</h1>
</body>
</html>
```

Output:-

Hello Javatpoint.com

3) CSS Class Selector

The class selector selects HTML elements with a specific class attribute. It is used with a period character . (full stop symbol) followed by the class name.

Example:-

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
   text-align: center;
   color: blue;
}
</style>
</head>
```

```
<body>
<h1 class="center">This heading is blue and center-aligned.</h1>
<p class="center">This paragraph is blue and center-aligned.</p>
</body>
</html>
```

Output:-

This heading is blue and center-aligned.

This paragraph is blue and center-aligned.

4) CSS Universal Selector

The universal selector is used as a wildcard character. It selects all the elements on the pages.

```
 * {

 margin: 0;

 padding: 0;

}
```

```
<!DOCTYPE html>
<html>
<head>
<style>
*{
  color: green;
  font-size: 20px;
}
</style>
</head>
<body>
<h2>This is heading</h2>
<p>This style will be applied on every paragraph.</p>
<p>Me too!</p>
<p>And me!</p>
</body>
</html>
```

Output:-

This is heading

This style will be applied on every paragraph.

Me too!

And me!

5) CSS Group Selector

The grouping selector is used to select all the elements with the same style definitions.

Grouping selector is used to minimize the code. Commas are used to separate each selector in grouping.

As you can see, you need to define CSS properties for all the elements. It can be grouped in following ways:

```
h1,h2,p {
   text-align: center;
   color: blue;
}
```

Example:-

```
<!DOCTYPE html>
<html>
<head>
<style>
h1, h2{
   text-align: center;
   color: blue;
}
</style>
</head>
<body>
<h1>Hello Javatpoint.com</h1>
<h2>Hello Javatpoint.com (In smaller font)</h2>
<p>This is a paragraph.</p>
</body>
</html>
```

Output:-

Hello Javatpoint.com

Hello Javatpoint.com (In smaller font)

This is a paragraph.

6	Write a Bootstrap program that demonstrates a 12-column responsive grid layout. <!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1"> <title>Bootstrap 12-Column Grid</title> <!-- Bootstrap CSS CDN --> <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.cs rel="stylesheet"> </head> <body> <h2 class="text-center mb-4">Bootstrap 12-Column Grid Layout</h2> <!-- Row 1: 12 columns each occupying 1 column width --> 1 2	[10]	CO3	L3

```
  <div class="col-1 bg-success text-white text-center p-2">3</div>

  <div class="col-1 bg-danger text-white text-center p-2">4</div>

  <div class="col-1 bg-warning text-dark text-center p-2">5</div>

  <div class="col-1 bg-info text-white text-center p-2">6</div>

  <div class="col-1 bg-dark text-white text-center p-2">7</div>

  <div class="col-1 bg-primary text-white text-center p-2">8</div>

  <div class="col-1 bg-secondary text-white text-center p-2">9</div>

  <div class="col-1 bg-success text-white text-center p-2">10</div>

  <div class="col-1 bg-danger text-white text-center p-2">11</div>

  <div class="col-1 bg-warning text-dark text-center p-2">12</div>

</div>



<!-- Row 2: 3 columns each occupying 4 column widths -->

<div class="row mb-3">

  <div class="col-4 bg-info text-white text-center p-3">col-4</div>

  <div class="col-4 bg-dark text-white text-center p-3">col-4</div>

  <div class="col-4 bg-primary text-white text-center p-3">col-4</div>

</div>



<!-- Row 3: 2 columns each occupying 6 column widths -->
```

	   col-6 col-6 <!-- Row 4: 1 column occupying all 12 column widths --> col-12 <!-- Bootstrap JS (Optional for layout) --> <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script> </body> </html>			
7	Explain the following Canvas Methods with Example. i) Line ii) Rectangle iii) Circle iv) Text v) Gradient **HTML Canvas Coordinates** In HTML canvas, the coordinates are very important because, with the understanding of these coordinates, we can easily draw any shapes or lines on the canvas. x-axis coordinates define the coordinates in horizontal and y-axis coordinates define the coordinates in vertical. We can understand this concept with different shapes. **Syntax** element.moveTo(x, y);	[10]	CO1	L2

element.lineTo(100, 150);

HTML Canvas Lines

how to make lines in different styles on Canvas in HTML. There are various methods used to draw a line on canvas like *beginPath()*, *moveTo()*, *lineTo()*, and *stroke()*. There are also various properties like lineWidth, strokeStyle, etc. can be used to give styles to a line with the help of JavaScript.

Canvas Methods for Drawing Lines

Syntax

```
canvas1.beginPath();
canvas1.moveTo( 0, 0 );
canvas1.lineTo( 120, 150 );
canvas1.stroke();
```

Methods

- **beginPath():** This method defines the new path(It should be called before defining a new shape (like a line, arc, or rectangle), especially if you don't want it connected to the previous drawing.)
- **moveTo(x,y): T**his method is used to define the starting point.
- **lineTo(x,y): T**his method is used to define the endpoint. Here, x specifies the x-axis coordinates(horizontal) and y specifies the y-axis coordinates(verticle) of the line's endpoint.
- **stroke():** This method is used to draw the line.

Example:-

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

<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
</script>
```

</body>
</html>

HTML Canvas Shapes facilitate different shapes, i.e., rectangles, triangles, lines, arcs, and curves to draw on the HTML Canvas.

For instance, to draw the line in the Canvas, the path will be used that implements the beginPath() method. Setting the path's starting point is achieved with moveTo(), while lineTo() establishes the endpoint. To render the line, we use the stroke() method, with the default stroke color being black.

Syntax

```
context.beginPath();
context.moveTo();
context.lineTo();
context.stroke();
```

HTML Canvas Rectangles

The **HTML Canvas Rectangles** facilitate the rect() method to draw rectangles on canvas. There are various attributes in the rect(x, y, width, height) method such as x and y defining the coordinates of the upper-left corner of the rectangle, width defining the width of the rectangle, and height defining the height of the rectangle.

fillRect() Method

The fillRect() method is used to draw a filled rectangle. The color is defined using the fillStyle property.

Syntax

```
context.fillRect( x, y, width, height );
```

Attributes

- **x**: The x-coordinate in fillRect is one of the parameters of the upper-left corner of the rectangle.
- **y**: The y-coordinate in fillRect is one of the parameters of the upper-left corner of the rectangle.
- **width**: The width of the rectangle, in pixels.
- **height**: The height of the rectangle, in pixels.

Example:-

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="200" height="100" style="border:1px
solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
// Create gradient
ctx.fillStyle = "yellow";
ctx.fillRect(10,10,150,80);
</script>
</body>
</html>
```

HTML Canvas Circles

HTML Canvas Circles facilitates the arc() method to make a circle where 0 is defined as the start angle and the end angle is 2*PI. The stroke() method is used to draw an outline of the circle and fill() is used to draw a filled circle we can give color with the fillStyle property.

```
<!DOCTYPE html>

<html>

<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#d3d3d3;">

Your browser does not support the HTML canvas tag.</canvas>

<script>

var c = document.getElementById("myCanvas");

var ctx = c.getContext("2d");

ctx.beginPath();

ctx.arc(95,50,40,0,2*Math.PI);

ctx.stroke();

</script>

</body>

</html>
```

stroke() Method

Creating circles on an HTML canvas using the **stroke** method to outline the circle. In the .arc() method there are parameters where x defines the x-coordinates of the center of the circle, y defines the y-coordinate of the center of the circle, r defines the radius of the circle, startAngle sets the start angle, and endAngle sets the end angle.

Example: The example shows circles on Canvas using the stroke method.

HTML Canvas Text

HTML Canvas Text facilitates the two different methods the fillText() method and the strokeText() method to draw text on canvas. The property font is used to define the size of the font and the font style.

To add color to the filled text, the fillStyle property is used while the strokeStyle property is used to color the outline of the stroke text. By default the color of the stroke is black.

There are 2 methods to draw the text on the HTML Canvas:

Table of Content

We will explore both methods, their syntax & parameters & understand them with the help of basic examples.

fillText() Method

The fillText() method in HTML Canvas is used to draw filled text on the canvas.

Syntax

```
context.fillText(text, x, y);
```

Parameters

- **text:** The string you want to draw on canvas.
- **x:** The x-axis coordinate of the point to the start draws the text, the value is in pixels.
- **y:** The y-axis coordinate of the point to the start draws the text, the value is in pixels.

Example:-

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="200" height="100" style="border:1px
solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World",10,50);
</script>
</body>
</html>
```

2. Stroke text:-

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

<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("Hello World",10,50);
</script>

</body>
</html>
```

HTML Canvas Gradients

HTML Canvas Gradients is used to give a gradient effect on Canvas with the help of JavaScript. The different shapes, such as rectangles, circles, lines, text, etc, can be filled with Gradients. To create a gradient on Canvas, we can use two techniques, Linear Gradient and Radial Gradient.

Linear Gradients on Canvas

The Linear gradient facilitates the createLinearGradient(x, y, x1, y1) method to draw the linear gradient from left to right. The **addColorStop() method** defines the color stops and the color. It can be between 0 to 1.

Syntax

.createLinearGradient(x, y, x1, y1);

Parameters

- **x**: The x-axis coordinates define the starting point.
- **y**: The y-axis coordinates define the starting point.
- **x1**: The x1-axis coordinates define the ending point.
- **y1**: The y1-axis coordinates defines the ending point.

Example:-

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="200" height="100" style="border:1px
solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
// Create gradient
var grd = ctx.createLinearGradient(0,0,200,0);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");
// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10,10,150,80);
</script>
</body>
</html>
```

8	What is Web API? Explain Web Storage API with Example. **What is Web API?** API stands for Application Programming Interface. An API is some kind of interface that includes a set of functions and subroutines that allow programmers to access specific features or data of an application, operating system or other services.	[10]	CO2	L1

A Web API is an application programming interface for the Web.

What is HTML Web Storage?

With web storage, applications can store data locally within the user's browser.

Before HTML5, application data had to be stored in cookies, included in every server request. Web storage is more secure, and large amounts of data can be stored locally, without affecting website performance.

Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.

Web storage is per origin (per domain and protocol). All pages, from one origin, can store and access the same data.

Web Storage API Objects

Web storage provides two objects for storing data in the browser:

- **window.localStorage** - stores data with no expiration date (data is not lost when the browser tab is closed)
- **window.sessionStorage** - stores data for one session (data is lost when the browser tab is closed)

Test Web Storage API Support

Before using web storage, we can quickly check browser support for localStorage and sessionStorage:

Example

```
<script>
const x = document.getElementById("result");
if (typeof(Storage) !== "undefined") {
  x.innerHTML = "Your browser supports Web storage!";
} else {
  x.innerHTML = "Sorry, no Web storage support!";
}
</script>
```

The localStorage Object

The localStorage object stores the data with no expiration date. The data will not be lost when the browser is closed, and will be available the next day, week, or year.

Example

Use localStorage to set and retrieve name and value pairs:

```
<script>
const x = document.getElementById("result");

if (typeof(Storage) !== "undefined") {
  // Store
  localStorage.setItem("lastname", "Smith");
  localStorage.setItem("bgcolor", "yellow");
  // Retrieve
  x.innerHTML = localStorage.getItem("lastname");
  x.style.backgroundColor = localStorage.getItem("bgcolor");
} else {
  x.innerHTML = "Sorry, no Web storage support!";
}
</script>
```

Example explained:

- Use the localStorage.setItem() method to create name/value pairs
- Use the localStorage.getItem() method to retrieve the values set
- Retrieve the value of "lastname" and insert it into an element with id="result"
- Retrieve the value of "bgcolor" and insert it into the style backgroundColor of the element with id="result”

The sessionStorage Object

The sessionStorage object is equal to the localStorage object, except that it stores the data for only one session! The data is deleted when the user closes the specific browser tab.

Counting Clicks with sessionStorage

The following example counts the number of times a user has clicked a button, in the current session:

Example

```
<script>
function clickCounter() {
  const x = document.getElementById("result");
  if (typeof(Storage) !== "undefined") {
    if (sessionStorage.clickcount) {
      sessionStorage.clickcount = Number(sessionStorage.clickcount)+1;
    } else {
     sessionStorage.clickcount = 1;
    }
    x.innerHTML = "You have clicked the button " +
sessionStorage.clickcount + " time(s) in this session!";
  } else {
    x.innerHTML = "Sorry, no Web storage support!";
  }
}
</script>
```

9	Explain Bootstrap Utilities with Example. ❖ **Bootstrap Utilities** **Definition:** **Bootstrap Utility Classes** are predefined helper classes used to **style, position, size, and align** HTML elements quickly **without writing custom CSS**. They are extremely useful for responsive and consistent UI design.	[10]	CO3	L3

🔹 Common Categories of Utilities in Bootstrap 5

Category	Purpose	Example
Display	Show/hide elements	`.d-none`, `.d-flex`, `.d-block`
Spacing	Margin and padding	`.m-3`, `.mt-2`, `.px-4`
Sizing	Width and height	`.w-25`, `.h-100`
Text	Text alignment, wrapping, color, etc.	`.text-center`, `.text-danger`
Flexbox	Flex utilities for layout	`.d-flex`, `.justify-content-between`
Positioning	Control position	`.position-relative`, `.top-0`
Colors	Text and background color	`.text-primary`, `.bg-light`
Borders	Add/remove borders, round corners	`.border`, `.rounded-pill`
Overflow	Control content overflow	`.overflow-auto`, `.overflow-hidden`
Visibility	Show/hide content responsively	`.visible`, `.invisible`, `.d-{breakpoint`

1. Spacing Utilities (Margin & Padding)

Definition:

Spacing utilities control **margin (m)** and **padding (p)** using shorthand notations.

🔸 **Syntax:**

m{side}-{breakpoint?}-{value}

p{side}-{breakpoint?}-{value}

m = margin, p = padding

Sides: t (top), b (bottom), s (start/left), e (end/right), x (left & right), y (top & bottom), blank (all sides)

Values: 0 to 5, auto

🔸 **Examples:**

Margin 3, Padding 2

Custom margins and paddings

2. Text Utilities

Definition:

Text utilities are used to **align text, transform case**, or **change text color**.

- **Syntax & Examples:**

Left-aligned primary text

Centered green text

Right-aligned red text

Uppercase Text

3. Display Utilities

Definition:

These control the **display behavior** of elements like block, inline, or hiding an element.

With Bootstrap 5 **display tools**, you can easily and quickly change the display value of components and more. With the help of Bootstrap responsive display utility classes, you may modify the value of the display property. For presentation purposes, we purposefully only support a portion of all potential values.

Use responsive display classes for showing and hiding items based on the device to speed up the construction of mobile-friendly websites. Instead of making completely distinct versions of the same website, responsively hide elements. Use the **.d-none** class or one of the **.d-{ sm, md, lg, xl, xxl }-none** classes for any responsive screen variation to simply hide components. You can combine one to have an element appear only on a specific range of screen sizes.

- **Syntax:**

Block Element

Inline Element

Hidden Element

- **Responsive Example:**

Visible on md and larger screens

4. Width and Height Utilities

Definition:

Used to set **width and height** using percentage-based classes.

- **Syntax:**

25% Width

100% Height

- **Example:**

Half Width Box

5. Border and Rounded Corners

Definition:

Utilities to add **borders and rounded corners** to elements.**Bootstrap 5 Borders** offer predefined border styles to elements. Classes like .border, .border-0, .border-top, .border-bottom, .border-left, and .border-right provide easy customization for borders, including color, width, and radius, enhancing UI design.

Property	Description
Border	**Classifies into additive and subtractive. Additive adds borders, subtractive removes.**
Border Color	**Utilizes theme colors to change border color.**
Border Width	**Increases or decreases border width.**
Border Radius	**Utilizes rounded classes for rounded corners.**

- **Syntax & Examples:**

Box with border

Rounded Box

- **6. Shadow Utilities**

Definition:

Used to add **shadow effects** to elements for a card-like or elevated look.

	🔸 **Syntax & Examples:** Small Shadow Large Shadow **7. Flexbox Utilities** **Flex** offers utility classes for creating flexible layouts using Flexbox, ensuring responsiveness across screen sizes. To start, wrap desired elements in a flex container with classes like .container or .row. Then, use flex utility classes to control item layout. **Definition:** Utility classes to **use and control flexbox layout**, alignment, and distribution. 🔸 **Syntax:** Item 1 Item 2 			
10	Explain the following SVG methods with Example. i) Ellipse ii) Polygon iii) Stroke iv) Fill v) Line ## SVG Ellipse - <ellipse> The <ellipse> element is used to create an ellipse. An ellipse is closely related to a circle. The difference is that an ellipse has an x and a y radius that differs from each other, while a circle has equal x and y radius. The <ellipse> element has four basic attributes to position and set the size of the ellipse: **Attribute** **Description** rx Required. The x radius of the ellipse ry Required. The y radius of the ellipse	[10]	CO2	L2

cx	The x-axis center of the ellipse. Default is 0
cy	The y-axis center of the ellipse. Default is 0

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

<h2>SVG ellipse Element</h2>

<svg height="140" width="500" xmlns="http://www.w3.org/2000/svg">
  <ellipse cx="120" cy="80" rx="100" ry="50"
style="fill:yellow;stroke:green;stroke-width:3" />
  Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

SVG Polygon - <polygon>

The <polygon> element is used to create a graphic that contains at least three sides.

Polygons are made of straight lines, and the shape is "closed" (it automatically connects the last point with the first).

The <polygon> element has one basic attribute that defines the points of the polygon:

Attribute	Description
points	Required. The list of points of the polygon. Each point must c coordinate and a y coordinate

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

<h2>SVG polygon Element</h2>

<svg height="220" width="500" xmlns="http://www.w3.org/2000/svg">
  <polygon points="100,10 150,190 50,190"
style="fill:lime;stroke:purple;stroke-width:3" />
  Sorry, your browser does not support inline SVG.
</svg>

</body>
```

```
</html>
```

SVG Polyline - <polyline>

The <polyline> element is used to create any shape that consists of only straight lines (that is connected at several points).

The <polyline> element has one basic attribute that defines the points of the polyline:

Attribute	Description
points	Required. The list of points of the polyline. Each point must c coordinate and a y coordinate

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

<h2>SVG polyline Element</h2>

<svg height="210" width="500" xmlns="http://www.w3.org/2000/svg">
  <polyline points="0,0 50,150 100,75 150,50 200,140 250,140"
style="fill:none;stroke:green;stroke-width:3" />
  Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

SVG Fill Attributes

The fill attribute sets the color of the inside of an element.

Here we will look at the three fill attributes:

- fill - sets the color of the inside of an element
- fill-opacity - sets the opacity of the color inside an element
- fill-rule - sets the algorithm used to determine the inside part of the shape

---

1. **SVG fill Attribute**

The fill attribute defines the color of the inside of an element.

The fill attribute can be used with the following SVG elements: <circle>, <ellipse>, <path>, <polygon>, <polyline>, <rect>, <text>, <textPath>, <tref> and <tspan>.

The value of the fill attribute can be a color name, rgb value or a hex value.

Here we use the fill attribute for a polygon, rectangle, circle and a text:

I love SVG!

Here is the SVG code:

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

<h2>SVG fill attribute</h2>

<svg width="600" height="220" xmlns="http://www.w3.org/2000/svg">
  <polygon points="50,10 0,190 100,190" fill="lime" />
  <rect width="150" height="100" x="120" y="50" fill="blue" />
  <circle r="45" cx="350" cy="100" fill="red" />
  <text x="420" y="100" fill="red">I love SVG!</text>
  Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

2. SVG fill-opacity Attribute

The fill-opacity attribute defines the opacity of the fill color.

The fill-opacity attribute can be used with the following SVG elements: <circle>, <ellipse>, <path>, <polygon>, <polyline>, <rect>, <text>, <textPath>, <tref> and <tspan>.

The value of the fill-opacity attribute goes from 0 to 1 (or 0% to 100%).

Here we use the fill-opacity attribute for a polygon, rectangle, circle and a text:

I love SVG!

Here is the SVG code:

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

```
<h2>SVG fill-opacity attribute</h2>

<svg width="600" height="220" xmlns="http://www.w3.org/2000/svg">
  <polygon points="50,10 0,190 100,190" fill="lime" fill-opacity="0.5" />
  <rect width="150" height="100" x="120" y="50" fill="blue" fill-opacity="50%"
/>
  <circle r="45" cx="350" cy="100" fill="red" fill-opacity="0.6" />
  <text x="420" y="100" fill="red" fill-opacity="70%">I love SVG!</text>
  Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

3. SVG fill-rule Attribute

The fill-rule attribute defines the algorithm used to determine the inside part of a shape.

The fill-rule attribute can be used with the following SVG elements: <path>, <polygon>, <polyline>, <text>, <textPath>, <tref> and <tspan>.

The value of the fill-rule attribute can be "nonzero" or "evenodd".

Here we use the fill-rule attribute for a polygon, with value "evenodd":

Here is the SVG code:

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

<h2>SVG fill-rule attribute</h2>

<svg height="210" width="500" xmlns="http://www.w3.org/2000/svg">
  <polygon points="100,10 40,198 190,78 10,78 160,198" fill="lime"
fill-rule="evenodd" />
  Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

Here we use the fill-rule attribute for a polygon, with value "nonzero":

Here is the SVG code:

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

<h2>SVG fill-rule attribute</h2>

<svg height="210" width="500" xmlns="http://www.w3.org/2000/svg">
  <polygon points="100,10 40,198 190,78 10,78 160,198" fill="lime"
fill-rule="nonzero" />
  Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

SVG Stroke Attributes

The stroke attribute sets the color of the line drawn around an element.

Here we will look at the six stroke attributes:

- stroke - sets the color of the line around an element
- stroke-width - sets the width of the line around an element
- stroke-opacity - sets the opacity of the line around an element
- stroke-linecap - sets the shape of the end-lines for a line or open path

SVG stroke Attribute

The stroke attribute defines the color of the outline of an element.

The stroke attribute can be used with the following SVG elements: <circle>, <ellipse>, <line>, <path>, <polygon>, <polyline>, <rect>, <text>, <textPath>, <tref> and <tspan>.

The value of the stroke attribute can be a color name, rgb value or a hex value.

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

<h2>SVG stroke attribute</h2>

<svg width="600" height="220" xmlns="http://www.w3.org/2000/svg">
  <polygon points="50,10 0,190 100,190" fill="lime" stroke="red" />
  <rect width="150" height="100" x="120" y="50" fill="yellow" stroke="red" />
  <circle r="45" cx="350" cy="100" fill="pink" stroke="blue" />
```

```
  <text x="420" y="100" fill="red" stroke="blue">I love SVG!</text>
  Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

SVG stroke-width Attribute

The stroke-width attribute defines the width of the stroke.

The stroke-width attribute can be used with the following SVG elements: <circle>, <ellipse>, <line>, <path>, <polygon>, <polyline>, <rect>, <text>, <textPath>, <tref> and <tspan>.

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

<h2>SVG stroke-width attribute</h2>

<svg width="600" height="220" xmlns="http://www.w3.org/2000/svg">
  <polygon points="55,10 10,190 110,190" fill="lime" stroke="red"
stroke-width="4" />
  <rect width="150" height="100" x="120" y="50" fill="yellow" stroke="red"
stroke-width="4" />
  <circle r="45" cx="350" cy="100" fill="pink" stroke="blue" stroke-width="4" />
  <text x="420" y="100" fill="red" stroke="blue" stroke-width="4">I love
SVG!</text>
  Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

SVG stroke-opacity Attribute

The stroke-opacity attribute defines the opacity of the stroke.

The stroke-opacity attribute can be used with the following SVG elements: <circle>, <ellipse>, <line>, <path>, <polygon>, <polyline>, <rect>, <text>, <textPath>, <tref> and <tspan>.

The value of the stroke-opacity attribute goes from 0 to 1 (or 0% to 100%).

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

```
<body>

<h2>SVG stroke-opacity attribute</h2>

<svg width="600" height="220" xmlns="http://www.w3.org/2000/svg">
  <polygon points="55,10 10,190 110,190" fill="lime" stroke="red"
stroke-width="4" stroke-opacity="0.4" />
  <rect width="150" height="100" x="120" y="50" fill="yellow" stroke="red"
stroke-width="4" stroke-opacity="0.4" />
  <circle r="45" cx="350" cy="100" fill="pink" stroke="blue" stroke-width="4"
stroke-opacity="0.4" />
  <text x="420" y="100" fill="red" stroke="blue" stroke-width="4"
stroke-opacity="0.4">I love SVG!</text>
  Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

Drawing a Line

The <line> element is used to create a line.
The <line> element creates a line between the start position (x1,y1) and the end position (x2,y2).
The <line> element has four basic attributes to position and set the length of the line:

Attribute	Description
x1	The start of the line on the x-axis
y1	The start of the line on the y-axis
x2	The end of the line on the x-axis
y2	The end of the line on the y-axis

```
<!DOCTYPE html>
<html>
<body>
<h2>SVG line Element</h2>
<svg height="200" width="300"
xmlns="http://www.w3.org/2000/svg">
  <line x1="0" y1="0" x2="300" y2="200"
style="stroke:red;stroke-width:2" />
  Sorry, your browser does not support inline SVG.
</svg>
```

	</body> </html>			