AWT SOLUTION

UNIT 1

Q.1.Explain Architecture of web browser.

ANS :-web Browser is an application software that allows us to view and explore information on the web. User can request for any web page by just entering a URL into address bar.

Web browser can show text, audio, video, animation and more. It is the responsibility of a web browser to interpret text and commands contained in the web page.
Architecture
There are a lot of web browser available in the market. All of them interpret and display information on the screen however their capabilities and structure varies depending upon implementation. But the most basic component that all web browser must exhibit are listed below:

Controller/Dispatcher

Interpreter

Client Programs

Controller works as a control unit in CPU. It takes input from the keyboard or mouse, interpret it and make other services to work on the basis of input it receives.

Interpreter receives the information from the controller and execute the instruction line by line. Some interpreter are mandatory while some are optional For example, HTML interpreter program is mandatory and java interpreter is optional.

Client Program describes the specific protocol that will be used to access a particular service. Following are the client programs tat are commonly used:

HTTP

SMTP

FTP

NNTP

POP

Q.2.Write a short note on web server.

ANS:-Web server is a program which processes the network requests of the users and serves them with files that create web pages. This exchange takes place using Hypertext Transfer Protocol (HTTP).
Basically, web servers are computers used to store HTTP files which makes a website and when a client requests a certain website, it delivers the requested website to the client. For example, you want to open Facebook on your laptop and enter the URL in the search bar of google. Now, the laptop will send an HTTP request to view the facebook webpage to another computer known as the webserver. This computer (webserver) contains all the files (usually in HTTP format) which make up the website like text, images, gif files, etc. After processing the request, the webserver will send the requested website-related files to your computer and then you can reach the website.
Different websites can be stored on the same or different web servers but that doesn’t affect the actual website that you are seeing in your computer. The web server can be any software or hardware but is usually

a software running on a computer. One web server can handle multiple users at any given time which is a necessity otherwise there had to be a web server for each user and considering the current world population, is nearly close to impossible. A web server is never disconnected from the internet because if it was, then it won't be able to receive any requests, and therefore cannot process them.

Q.3.What is HTTP? Explain how browser and server communicate using HTTP request and response.
OR
Discuss HTTP. Explain HTTP request and HTTP response mechanism over the Internet.

ANS:-HTTP stands for Hyper Text Transfer Protocol.
Communication between client computers and web servers is done by sending HTTP Requests and receiving HTTP Responses
The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems.
Basically, HTTP is a TCP/IP based communication protocol, that is used to deliver data (HTML files, image files, query results, etc.) on the World Wide Web.
HTTP Request / Response
Communication between clients and servers is done by requests and responses:

-A client (a browser) sends an HTTP request to the web

A web server receives the request
1.The server runs an application to process the request
2.The server returns an HTTP response (output) to the browser
3.The client (the browser) receives the response
4.The HTTP Request Circle

-A typical HTTP request / response circle:

1.The browser requests an HTML page. The server returns an HTML file.
2.The browser requests a style sheet. The server returns a CSS file.
3.The browser requests an JPG image. The server returns a JPG file.
4.The browser requests JavaScript code. The server returns a JS file
5.The browser requests data. The server returns data (in XML or JSON).

-XHR - XML Http Request

All browsers have a built-in XMLHttpRequest Object (XHR).

XHR is a JavaScript object that is used to transfer data between a web browser and a web server.

XHR is often used to request and receive data for the purpose of modifying a web page.

Despite the XML and Http in the name, XHR is used with other protocols than HTTP, and the data can be of many different types like HTML, CSS, XML, JSON, and plain text.

Q.4.What is full form of CORS? Why are CORS needed?

ANS:-CORS is called as Cross-Origin Resource Sharing.
Cross-origin resource sharing (CORS) is a security relaxation measure that needs to be implemented in some APIs in order to let web browsers access them. However, when CORS is enabled by a back-end developer some security analysis needs to be done in order to ensure you're not relaxing your server security too much. The problem usually arises when you allow resource sharing for every resource rather than for just specific ones.

We'll take a look at some of the security risks of implementing CORS. First, it's important to get a basic understanding of what CORS is, how it works, and how you can use security relaxation measures without leaving yourself vulnerable.

Q.5.Explain the term
1)WWW
2)Website
3)XML

ANS:-

1.WWW:-
World Wide Web, which is also known as a Web, is a collection of websites or web pages stored in web servers and connected to local computers through the internet. These websites contain text pages, digital images, audios, videos, etc. Users can access the content of these sites from any part of the world over the internet using their devices such as computers, laptops, cell phones, etc. The WWW, along with internet, enables the retrieval and display of text and media to your device.

A web page is given an online address called a Uniform Resource Locator (URL). A particular collection of web pages that belong to a specific URL is called a website, e.g., www.facebook.com, www.google.com, etc. So, the World Wide Web is like a huge electronic book whose pages are stored on multiple servers across the world.

Small websites store all of their WebPages on a single server, but big websites or organizations place their WebPages on different servers in different countries so that when users of a country search their site they could get the information quickly from the nearest server.

2.website:-
A website is a collection of many web pages, and web pages are digital files that are written using HTML(HyperText Markup Language). To make your website available to every person in the world, it must be stored or hosted on a computer connected to the Internet round a clock. Such computers are known as a Web Server.

The website's web pages are linked with hyperlinks and hypertext and share a common interface and design. The website might also contain some additional documents and files such as images, videos, or other digital assets.

With the Internet invading every sphere, we see websites for all kinds of causes and purposes. So, we can also say that a website can also be thought of as a digital environment capable of delivering information and solutions and promoting interaction between people, places, and things to support the goals of the organization it was created for.

Components of a Website:
1.Webhost
2.Address
3.Homepage
4.Design
5.Content
6.The Navigation Structure

3.XML:-
Xml (eXtensible Markup Language) is a mark up language.
XML is designed to store and transport data.
Xml was released in late 90's. it was created to provide an easy to use and store self describing data.
XML became a W3C Recommendation on February 10, 1998.
XML is not a replacement for HTML.

XML is designed to be self-descriptive.
XML is designed to carry data, not to display data.
XML tags are not predefined. You must define your own tags.
XML is platform independent and language independent.

XML stands for extensible markup language. A markup language is a set of codes, or tags, that describes the text in a digital document. The most famous markup language is hypertext markup language (HTML), which is used to format Web pages. XML, a more flexible cousin of HTML, makes it possible to conduct complex business over the Internet.

Q.6.Explain Web security in detail.

ANS:-
Web Security is very important nowadays. Websites are always prone to security threats/risks. Web Security deals with the security of data over the internet/network or web or while it is being transferred to the internet. For e.g. when you are transferring data between client and server and you have to protect that data that security of data is your web security.

Hacking a Website may result in the theft of Important Customer Data, it may be the credit card information or the login details of a customer or it can be the destruction of one's business and propagation of illegal content to the users while somebody hacks your website they can either steal the important information of the customers or they can even propagate the illegal content to your users through your website so, therefore, security considerations are needed in the context of web security.

Security Threats:
A Threat is nothing but a possible event that can damage and harm an information system. Security Threat is defined as a risk that which, can potentially harm Computer systems & organizations. Whenever an Individual or an Organization creates a website, they are vulnerable to security attacks.

Security attacks are mainly aimed at stealing altering or destroying a piece of personal and confidential information, stealing the hard drive space, and illegally accessing passwords. So whenever the website you created is vulnerable to security attacks then the attacks are going to steal your data alter your data destroy your personal information see your confidential information and also it accessing your password.

Top Web Security Threats :
Web security threats are constantly emerging and evolving, but many threats consistently appear at the top of the list of web security threats. These include:

Cross-site scripting (XSS)
SQL Injection
Phishing
Ransomware
Code Injection
Viruses and worms
Spyware
Denial of Service

Security Consideration:
1.Updated Software
2.Beware of SQL Injection
3.Cross-Site Scripting (XSS)
4.Error Messages
5.Data Validation
6.Password

Q.7.Differentiate between GET and POST method.

ans:-

get:-
- In case of Get request, only limited amount of data can be sent because data is sent in header.
- Get request is not secured because data is exposed in URL bar.
- Get request can be bookmarked.
- Get request is idempotent . It means second request will be ignored until response of first request is delivered
- Get request is more efficient and used more than Post.
- In GET method only ASCII characters are allowed. I

post:-
- In case of post request, large amount of data can be sent because data is sent in body.
- Post request is secured because data is not exposed in URL bar.
- Post request cannot be bookmarked.
- Post request is non-idempotent.
- Post request is less efficient and used less than get.
- In POST method all types of data is allowed.

UNIT 2

Q.8.Explain the basic structure of HTML documents. (webpage structure).

The basic structure of an HTML document consists of 5 elements:

1.<!DOCTYPE>
2.<html>
3.<head>
4.<meta>
5.<title>
6.<body>

~<!DOCTYPE>:-

A DOCTYPE declaration must be specified on the first line of each web document:
doctypeThe DOCTYPE tells the web browser which version of HTML the page is written in. In this class, we will be using 'XHTML Transitional', which allows us a little flexibility.

~The <html> Element:-

Immediately following the DOCTYPE declaration is the <html> element:
The <html> element tells the browser that the page will be formatted in HTML and, optionally, which world language the page content is in.

~The <head> and <body> Element:-

The <head> element surrounds all the special "behind the scenes" elements of a web document. Most of these elements do not get displayed directly on the web page.
The <body> element surrounds all the actual content (text, images, videos, links, etc.) that will be displayed on our web page.

~The <meta> Element:-

Immediately after the <head> line, we place this <meta> element:meta-elementThis line declares that the document is encoded in the UTF-8 (Unicode) character set.

There can be multiple <meta> lines in the same web page. The <meta> element is often used to provide additional information such as page keywords, a page description, and the author(s) of a web document.

~The <title> Element:-

The <title> element defines what text will show in the web browser's title bar:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
</head>
<body>

</body>
</html>
```

Q.9.Explain ordered list and unordered list in HTML with proper example.

ordered list:-

An ordered list defines a list of items in which the order of the items are matters. An ordered list is also called a number list. The ordering is given by a numbering scheme, using Arabic numbers, letters, roman numerals. Or in other words, ordered list tag is used to create ordered list.

Syntax:

 content

Attributes of ordered list:

1. reversed: This attribute is used to specifies that the ordered of the list should be reversed.

Syntax:

<ol reversed>

2. start: This attribute is used to specifies the start value of the list.

Syntax:

<ol start = "number">

3. type: This attribute is used to specifies the type of list item maker. The value of this attribute is decimal(Default)/lower-roman/upper roman/lower-alpha/upper alpha

Syntax:

<ol type = "1|b|A|i|I">

example:-

```
<html>
<head>
        <title>ordered list</title>
```

```
</head>
<body>
	<h1>Example of ordered list whose type = "A"</h1>
	<ol type="A">
		<li>Sachin</li>
		<li>Manoj</li>

	</ol>
	<h1>Example of reverse ordered list</h1>
	<ol reversed>
		<li>Parth</li>
		<li>sujay</li>
	</ol>
	<h1>Example of ordered list start from 10</h1>
	<ol start = "10">
		<li>Pushpa</li>
		<li>Purvi</li>
	</ol>

</body>
</html>
```

output:-

Example of ordered list whose type = "A"
A.Sachin
B.Manoj
Example of reverse ordered list
2.Parth
1.sujay
Example of ordered list start from 10
10.Pushpa
11.Purvi

unordered list:-

An unordered list defines a list of items in which the order of the items does not. Or in other words, unordered list tag is used to create a unordered list. It is also known as bulleted list. In unordered list each element in the list is defined using <li> tag.

Syntax:

 content

Attributes of unordered list:

List-style-type: This attribute is used to specifies the bullet style that will be used as the list item marker. The value of this attribute is None/disc(default)/circle/square.

Syntax:

<ul style=”list-style-type:square|disc|none;”>

example:-

```
<html>
<head>
```

```
        <title>unordered list</title>
</head>
<body>
        <h1>Example of unordered list in default</h1>
        <ul>
                <li>Sachin</li>
                <li>Manoj</li>
                <li>Parth</li>
                <li>sujay</li>
                <li>Amraditya</li>
        </ul>
</body>
</html>
```

output:-

Example of unordered list in default
.Sachine
.Manoj
.Parth
.sujay
.Amraditya

Q.10.Explain following html tags with proper example.
<img> , <a>, <select>, ,<input>

ANS:-

~<img>

The <img> tag is used to embed an image in an HTML page.

Images are not technically inserted into a web page; images are linked to web pages. The <img> tag creates a holding space for the referenced image.

The <img> tag has two required attributes:

src - Specifies the path to the image
alt - Specifies an alternate text for the image, if the image for some reason cannot be displayed
Note: Also, always specify the width and height of an image. If width and height are not specified, the page might flicker while the image loads.

The <img> tag also supports the Global Attributes in HTML.
The <img> tag also supports the Event Attributes in HTML.

```
<img src="img_girl.jpg" alt="Girl in a jacket" width="500" height="600">
```

~<a>

The <a> tag defines a hyperlink, which is used to link from one page to another.

The most important attribute of the <a> element is the href attribute, which indicates the link's destination.

By default, links will appear as follows in all browsers:

An unvisited link is underlined and blue

A visited link is underlined and purple
An active link is underlined and red

If the <a> tag has no href attribute, it is only a placeholder for a hyperlink.
A linked page is normally displayed in the current browser window, unless you specify another target.
Use CSS to style links: CSS Links and CSS Buttons.

The <a> tag also supports the Global Attributes in HTML.
The <a> tag also supports the Event Attributes in HTML.

Visit W3Schools.com!

~<select>

The <select> element is used to create a drop-down list.

The <select> element is most often used in a form, to collect user input.

The name attribute is needed to reference the form data after the form is submitted (if you omit the name attribute, no data from the drop-down list will be submitted).

The id attribute is needed to associate the drop-down list with a label.

The <option> tags inside the <select> element define the available options in the drop-down list.

: Always add the <label> tag for best accessibility practices!

The <select> tag also supports the Global Attributes in HTML.
The <select> tag also supports the Event Attributes in HTML.

example:-
<label for="cars">Choose a car:</label>

```
<select name="cars" id="cars">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="mercedes">Mercedes</option>
  <option value="audi">Audi</option>
</select>
```

~

The tag is an inline container used to mark up a part of a text, or a part of a document.

The tag is easily styled by CSS or manipulated with JavaScript using the class or id attribute.

The tag is much like the element, but is a block-level element and is an inline element.

The tag also supports the Global Attributes in HTML.
The tag also supports the Event Attributes in HTML.

example:-
My mother has blue eyes.

~<input>

The <input> tag specifies an input field where the user can enter data.

The <input> element is the most important form element.

The <input> element can be displayed in several ways, depending on the type attribute.

The different input types are as follows:

<input type="button">
<input type="checkbox">
<input type="color">
<input type="date">
<input type="datetime-local">
<input type="email">
<input type="file">
<input type="hidden">
<input type="image">
<input type="month">
<input type="number">
<input type="password">
<input type="radio">
<input type="range">
<input type="reset">
<input type="search">
<input type="submit">
<input type="tel">
<input type="text"> (default value)
<input type="time">
<input type="url">
<input type="week">

Always use the <label> tag to define labels for <input type="text">, <input type="checkbox">, <input type="radio">, <input type="file">, and <input type="password">.

The <input> tag also supports the Global Attributes in HTML.
The <input> tag also supports the Event Attributes in HTML.

example:-

```
<form action="/action_page.php">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="submit" value="Submit">
</form>
```

Q.12.What is rowspan and colspan used in HTML? Give examples.

ANS:-

~rowspan:-
The rowspan attribute in HTML specifies the number of rows a cell should span. That is if a row spans two rows, it means it will take up the space of two rows in that table. It allows the single table cell to span the

height of more than one cell or row. It provides the same functionality as “merge cell” in the spreadsheet program like Excel.
Usage: It can be used with <td> and <th> element in an HTML Table.

Attribute Values: It contains a value i.e number Which specify the number of rows that a table cell should span.

<td>: The rowspan attribute when used with <td> tag determines the number of standard cells it should span.
Syntax:
<td rowspan = "value">table content...</td>

~colspan:-
The colspan attribute in HTML specifies the number of columns a cell should span. It allows the single table cell to span the width of more than one cell or column. It provides the same functionality as “merge cell” in a spreadsheet program like Excel.
Usage: It can be used with <td> and <th> element while creating an HTML Table.

Attribute Values: It contains a value i.e number Which specify the number of columns that a cell should span.

Note: colspan=”0″ tells the browser to span the cell to the last column of the column group (colgroup).

<td>: The colspan attribute when used with <td> tag determines the number of standard cells it should span.
Syntax:
<td colspan = "value">table content...</td>

Q.13.What is meta tag? How it is useful by search engine? OR Explain all the Meta Tags with example.

ANS:-

The <meta> tag defines metadata about an HTML document. Metadata is data (information) about data.
<meta> tags always go inside the <head> element, and are typically used to specify character set, page description, keywords, author of the document, and viewport settings.
Metadata will not be displayed on the page, but is machine parsable.
Metadata is used by browsers (how to display content or reload page), search engines (keywords), and other web services.
There is a method to let web designers take control over the viewport (the user's visible area of a web page), through the <meta> tag

Meta tags are snippets of code that tell search engines important information about your web page, such as how they should display it in search results. They also tell web browsers how to display it to visitors.
Every web page has meta tags, but they’re only visible in the HTML code.

all meta tag with example:-

1. <meta charset="utf-8">
It defines the character encoding. The value of charset is "utf-8" which means it will support to display any language.

2. <meta name="keywords" content="HTML, CSS, JavaScript, Tutorials">
It specifies the list of keyword which is used by search engines.

3. <meta name="description" content="Free Online tutorials">
It defines the website description which is useful to provide relevant search performed by search engines.

4. <meta name="author" content="thisauthor">
It specifies the author of the page. It is useful to extract author information by Content management system automatically.

5. <meta name="refresh" content="50">
It specifies to provide instruction to the browser to automatically refresh the content after every 50sec (or any given time).

6. <meta http-equiv="refresh" content="5; url=https://www.javatpoint.com/html-tags-list">
In the above example we have set a URL with content so it will automatically redirect to the given page after the provided time.

7. <meta name="viewport" content="width=device-width, initial-scale=1.0">
It specifies the viewport to control the page dimension and scaling so that our website looks good on all devices. If this tag is present, it indicates that this page is mobile device supported.

Q.14.Demonstrate the use of Frame and Frameset with proper example.

frameset:-
The <frameset> tag in HTML is used to define the frameset. The <frameset> element contains one or more frame elements. It is used to specify the number of rows and columns in frameset with their pixel of spaces. Each element can hold a separate document.
The <frameset> tag is not supported in HTML5.

syntex:-
<frameset cols = "pixels|%|*">

frame:-
HTML Frames are used to divide the web browser window into multiple sections where each section can be loaded separately. A frameset tag is the collection of frames in the browser window.
Instead of using body tag, use frameset tag in HTML to use frames in web browser.

example:-

```
<html>
   <head>
      <title>
         Frame tag
      </title>
   </head>
   <frameset rows="20%,*,20%">
      <frame src="header.html">
         <frameset cols="20%,*,20%">
            <frame src="left.html">
            <frame src="main.html" name="main">
            <frame src="right.html">
         </frameset>
      <frame src="footer.html">
   </frameset>
</html>
```

Q.15.Design Registration form and do proper validation with HTML 5 inbuilt functionality. OR Explain Different Form elements/Form Controls.

ANS:-
<!DOCTYPE html>

```
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta http-equiv="refresh" content="60">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>practical2</title>
</head>

<body>
   <marquee behavior="alternate" direction=""><a href="Resume.html" target="_blank"> Click There For
My resume</a>
   </marquee>
   <form action="" method="post">
      <h1 align="center">Registration Form</h1>
      <hr>
      <table border="2" width="50%" height="50%" align="center" >
         <tr>
            <th><b>First Name :</b></th>
            <td><input type="text" required placeholder="Enter First Name" maxlength="50"></td>
         </tr>

         <tr>
            <th> <b>Last Name : </b></th>
            <td><input type="text" required placeholder="Enter Last Name" maxlength="50"></td>
         </tr>

         <tr>
            <th> <b>Contact no. :</b></th>
            <td><input type="text" pattern="\d*" minlength="10" maxlength="10" required
placeholder="Enter Your Contact No."></td>
         </tr>

         <tr>
            <th> <b>Landline no. :</b></th>
            <td><input type="text" pattern="\d*{3}-\d*{6}" maxlength="10" required
                  placeholder="Enter Your Landline No.">
         </tr>

         <tr>
            <th><b>Email :</b></th>
            <td><input type="email" required placeholder="Enter Your Email ID"></td>
         </tr>

         <tr>
            <th>Password</th>
            <td><input type="password" placeholder="Enter a New Password"></td>
         </tr>

         <tr>
            <th><b>Date of Birth</b></th>
            <td><input type="date" required></td>
         </tr>

         <tr>
            <th>Age</th>
```

```
            <td><input type="Number" min="18" max="40" placeholder="Enter Age Between 18 to 40"
required></td>
        </tr>
        <tr>
            <th> <b>Gender</b></th>
            <td>Male
                <input type="radio" name="gender">
                Female
                <input type="radio" name="gender">
            </td>
        </tr>

        <tr>
            <th>Address</th>
            <td><textarea name="Address" id="" cols="60" rows="3"></textarea></td>

        </tr>

        <tr>
            <th>City</th>
            <td><select name="City" id="">
                    <option value="" disabled selected>Select City</option>
                    <option value="">Ahmedabad</option>
                    <option value="">Vadodara</option>
                    <option value="">Morbi</option>
                </select>
            </td>
        </tr>

        <tr>
            <th>Pincode</th>
            <td><input type="text" pattern="\d*" maxlength="6"></td>
        </tr>

        <tr>
            <th><b>Joining Date</b></th>
            <td><input type="date" required max="2022-12-31" min="2022-01-01"></td>
        </tr>

        <tr>
            <th>Cnfirm Your Form</th>
            <td> <input type="submit"> <input type="reset"></td>
        </tr>
    </table>

  </form>


</body>

</html>
```

The HTML <form> element can contain one or more of the following form elements:

<input>
<label>

<select>
<textarea>
<button>
<fieldset>
<legend>
<datalist>
<output>
<option>
<optgroup>

The <input> Element

One of the most used form element is the <input> element.

The <input> element can be displayed in several ways, depending on the type attribute.

Example

```
<label for="fname">First name:</label>
<input type="text" id="fname" name="fname">
```

The <label> Element

The <label> element defines a label for several form elements.

The <label> element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focus on the input element.

The <select> Element

The <select> element defines a drop-down list:
The <option> elements defines an option that can be selected.

By default, the first item in the drop-down list is selected.

Example

```
<label for="cars">Choose a car:</label>
<select id="cars" name="cars">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

The <textarea> Element

The <textarea> element defines a multi-line input field (a text area):

Example

```
<textarea name="message" rows="10" cols="30">
The cat was playing in the garden.
</textarea>
```

The rows attribute specifies the visible number of lines in a text area.
The cols attribute specifies the visible width of a text area.

The <button> Element

The <button> element defines a clickable button:

Example

```
<button type="button" onclick="alert('Hello World!')">Click Me!</button>
```

The <fieldset> and <legend> Elements

The <fieldset> element is used to group related data in a form.

The <legend> element defines a caption for the <fieldset> element.

Example

```
<form action="/action_page.php">
  <fieldset>
    <legend>Personalia:</legend>
    <label for="fname">First name:</label><br>
    <input type="text" id="fname" name="fname" value="John"><br>
    <label for="lname">Last name:</label><br>
    <input type="text" id="lname" name="lname" value="Doe"><br><br>
<input type="submit" value="Submit">
  </fieldset>
</form>
```

Q.16.Discuss SEO.

ANS:-
SEO means Search Engine Optimization and is the process used to optimize a website's technical configuration, content relevance and link popularity so its pages can become easily findable, more relevant and popular towards user search queries, and as a consequence, search engines rank them better.

Search engines recommend SEO efforts that benefit both the user search experience and page's ranking, by featuring content that fulfills user search needs. This includes the use of relevant keywords in titles, meta descriptions, and headlines (H1), featuring descriptive URLs with keywords rather than strings of numbers, and schema markup to specify the page's content meaning, among other SEO best practices.

Search engines help people find what they're looking for online. Whether researching a product, looking for a restaurant, or booking a vacation, search engines are a common starting point when you need information. For business owners, they offer a valuable opportunity to direct relevant traffic to your website.

Search engine optimization (SEO) is the practice of orienting your website to rank higher on a search engine results page (SERP) so that you receive more traffic. The aim is typically to rank on the first page of Google results for search terms that mean the most to your target audience. So, SEO is as much about understanding the wants and needs of your audience as it is about the technical nature of how to configure your website.

Search engines provide results for any search query a user enters. To do so, they survey and "understand" the vast network of websites that make up the web. They run a sophisticated algorithm that determines what results to display for each search query.

UNIT 3

Q.17.What is CSS? What are the benefits of CSS? List out the different ways to write CSS.
OR
List and Explain types of CSS with Example.

ANS:-
Cascading Style Sheet(CSS) is used to set the style in web pages that contain HTML elements. It sets the background color, font-size, font-family, color, … etc property of elements on a web page.

There are three types of CSS which are given below:

1.Inline CSS
2.Internal or Embedded CSS
3.External CSS

~Inline CSS:
Inline CSS contains the CSS property in the body section attached with element is known as inline CSS. This kind of style is specified within an HTML tag using the style attribute.
Example:

```
<!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>
```

~Internal or Embedded CSS:
This can be used when a single HTML document must be styled uniquely. The CSS rule set should be within the HTML file in the head section i.e the CSS is embedded within the HTML file.
Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Internal CSS</title>
      <style>
         .main {
            text-align:center;
         }
         .GFG {
            color:#009900;
            font-size:50px;
            font-weight:bold;
         }
         .geeks {
            font-style:bold;
            font-size:20px;
         }
      </style>
   </head>
   <body>
      <div class = "main">
         <div class ="GFG">GeeksForGeeks</div>

         <div class ="geeks">
            A computer science portal for geeks
         </div>
      </div>
   </body>
</html>
```

~External CSS:
External CSS contains separate CSS file which contains only style property with the help of tag attributes (For example class, id, heading, … etc). CSS property written in a separate file with .css extension and should be linked to the HTML document using link tag. This means that for each element, style can be set only once and that will be applied across web pages.
Example: The file given below contains CSS property. This file save with .css extension. For Ex: geeks.css

```
body {
    background-color:powderblue;
}
.main {
    text-align:center;
}
.GFG {
    color:#009900;
    font-size:50px;
    font-weight:bold;
}
#geeks {
    font-style:bold;
    font-size:20px;
}
```

Q.18.Explain Class and ID Selector in CSS with examples.

ANS:-
In CSS, class and ID selectors are used to identify various HTML elements. The main benefit of setting class or ID is that you can present the same HTML element differently, depending on its class or ID.

~Class selector
The class selector selects elements with a specific class attribute. It matches all the HTML elements based on the contents of their class attribute. The . symbol, along with the class name, is used to select the desired class.

Syntax

```
.class-name {
  /* Define properties here */
}
```

example:-

```
<!DOCTYPE html>
<html>
<head>
<style>
.example {
text-align: center;
color: blue;
font-size: 25px;
}
</style>
</head>
<body>
<h1 class="example">This heading is blue and center-aligned.</h1>
<p class="example">This paragraph is blue and center-aligned.</p>
```

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

~ID selector
The ID selector matches an element based on the value of its id attribute. In order for the element to be selected, its ID attribute must exactly match the value given in the selector. The # symbol and the id of the HTML element name are used to select the desired element.

Syntax

```
#idname {
  /* Define properties here */
}
```

example:-

```
<!DOCTYPE html>
<html>
<head>
<style>
#para {
text-align: center;
color: blue;
font-size: 25px;
background-color: pink;
}
</style>
</head>
<body>
<h1> Welcome to the Javatpoint.com </h1>
<p id = "para">This paragraph will be affected.</p>
<p>This paragraph will not be affected.</p>
</body>
</html>
```

Q.19.Demonstrate CSS Pseudo classes with example.

ANS:-
A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- .Style an element when a user mouses over it
- .Style visited and unvisited links differently
- .Style an element when it gets focus

Syntax
The syntax of pseudo-classes:

```
selector:pseudo-class {
  property: value;
}
```

~ Anchor Pseudo-classes
Links can be displayed in different ways:

Example

```
/* unvisited link */
a:link {
  color: #FF0000;
}

/* visited link */
a:visited {
  color: #00FF00;
}

/* mouse over link */
a:hover {
  color: #FF00FF;
}

/* selected link */
a:active {
  color: #0000FF;
}
```

Q.20.What is positioning in CSS? Explain it with example.

ANS:-
The position Property
The position property specifies the type of positioning method used for an element.

There are five different position values:

static
relative
fixed
absolute
sticky
Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

~position: static;
HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

This element has position: static;
Here is the CSS that is used:

Example
```
div.static {
  position: static;
  border: 3px solid #73AD21;
}
```

~position: relative;
An element with position: relative; is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

This element has position: relative;
Here is the CSS that is used:

Example
```
div.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
}
```
ADVERTISEMENT

~position: fixed;
An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been located.

Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:

Example
```
div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
```
This element has position: fixed;

~position: absolute;
An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

Note: Absolute positioned elements are removed from the normal flow, and can overlap elements.

Here is a simple example:

This element has position: relative;This element has position: absolute;
Here is the CSS that is used:

Example
```
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
}

div.absolute {
  position: absolute;
```

```
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
```
~position: sticky;
An element with position: sticky; is positioned based on the user's scroll position.

A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

Internet Explorer does not support sticky positioning. Safari requires a -webkit- prefix (see example below). You must also specify at least one of top, right, bottom or left for sticky positioning to work.

In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.

Example
```
div.sticky {
  position: -webkit-sticky; /* Safari */
  position: sticky;
  top: 0;
  background-color: green;
  border: 2px solid #4CAF50;
}
```

Q.21.Explain the use of Media Query in CSS.

ANS:-
Media queries are useful when you want to modify your site or app depending on a device's general type (such as print vs. screen) or specific characteristics and parameters (such as screen resolution or browser viewport width).

Media queries are used for the following:

To conditionally apply styles with the CSS @media and @import at-rules.
To target specific media for the <style>, <link>, <source>, and other HTML elements with the media= attribute.
To test and monitor media states using the Window.matchMedia() and MediaQueryList.addListener() JavaScript methods.
Note: The examples on this page use CSS's @media for illustrative purposes, but the basic syntax remains the same for all types of media queries.

Syntax
A media query is composed of an optional media type and any number of media feature expressions, which may optionally be combined in various ways using logical operators. Media queries are case-insensitive.

Media types define the broad category of device for which the media query applies: all, print, screen. The type is optional (assumed to be all) except when using the not or only logical operators.
Media features describe a specific characteristic of the user agent, output device, or environment: any-hover, any-pointer, aspect-ratio, color, color-gamut, color-index, device-aspect-ratio Deprecated, device-height Deprecated, device-width Deprecated, display-mode, dynamic-range, forced-colors, grid, height, hover, inverted-colors, monochrome, orientation, overflow-block, overflow-inline, pointer, prefers-color-scheme, prefers-contrast, prefers-reduced-motion, resolution, scripting, update, video-dynamic-range, width. For example, the hover feature allows a query to test against whether the device supports hovering over elements.

Media feature expressions test for their presence or value, and are entirely optional. Each media feature expression must be surrounded by parentheses.
Logical operators can be used to compose a complex media query: not, and, and only. You can also combine multiple media queries into a single rule by separating them with commas.
A media query computes to true when the media type (if specified) matches the device on which a document is being displayed and all media feature expressions compute as true. Queries involving unknown media types are always false.

Q.22.Explain CSS box model (border, margin, padding).

ANS:-
All HTML elements can be considered as boxes.

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
The box model allows us to add a border around elements, and to define space between elements.

Example
Demonstration of the box model:

```
div {
  width: 300px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
```

~Width and Height of an Element
In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

Important: When you set the width and height properties of an element with CSS, you just set the width and height of the content area. To calculate the full size of an element, you must also add padding, borders and margins.

Example
This element will have a total width of 350px:

```
div {
  width: 320px;
  padding: 10px;
  border: 5px solid gray;
  margin: 0;
}
```

Q.23.Demonstrate the use of CSS for Colors and Background.

ANS:-
~CSS background-color Property

Example
Set the background color for a page:
body {background-color: coral;}

~Definition and Usage
The background-color property sets the background color of an element.
The background of an element is the total size of the element, including padding and border (but not the margin).
Use a background color and a text color that makes the text easy to read.

CSS Syntax
background-color: color|transparent|initial|inherit;

~Example
Specify the background color with a HEX value:
body {background-color: #92a8d1;}

~Example
Specify the background color with an RGB value:
body {background-color: rgb(201, 76, 76);}

~Example
Specify the background color with an RGBA value:
body {background-color: rgba(201, 76, 76, 0.3);}

~Example
Specify the background color with a HSL value:
body {background-color: hsl(89, 43%, 51%);}

~Example
Specify the background color with a HSLA value:
body {background-color: hsla(89, 43%, 51%, 0.3);}

~Example
Set background colors for different elements:

```
body {
  background-color: #fefbd8;
}

h1 {
  background-color: #80ced6;
}

div {
  background-color: #d5f4e6;
}

span {
  background-color: #f18973;
}
```

UNIT 4

Q.24.What is JavaScript? What are the benefits of JavaScript?

ANS:-
JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of the web pages, whose implementation allows a client-side script to interact with a user and to make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

Fast speed: JavaScript is executed on the client side that's why it is very fast.
Easy to learn: JavaScript is easy to learn. Any one which have basic knowledge of programming can easily lean JavaScript.
Versatility: It refers to lots of skills. It can be used in a wide range of applications.
Browser Compatible: JavaScript supports all modern browsers. It can execute on any browser and produce same result.
Server Load: JavaScript reduce the server load as it executes on the client side.
Rich interfaces: JavaScript provides the drag and drop functionalities which can provides the rich look to the web pages.
Popularity: JavaScript is a very popular web language because it is used every where on the web.
Regular Updates: JavaScript updated annually by ECMA.
Popularity:Since all modern browsers support JavaScript, it is seen almost everywhere. All the famous companies use JavaScript as a tool including Google, Amazon, PayPal, etc.

Q.25.Differentiate Client side scripting and Server side scripting.

ANS:-
Client-side scripting:-

~Source code is visible to the user.
~Its main function is to provide the requested output to the end user.
~It usually depends on the browser and its version.
~It runs on the user's computer.
~There are many advantages linked with this like faster.
response times, a more interactive application.
~It does not provide security for data.
~It is a technique used in web development in which scripts run on the client's browser.
~HTML, CSS, and javascript are used.
~No need of interaction with the server.
~It reduces load on processing unit of the server.

Server side scripting:-

~Source code is not visible to the user because its output
of server-sideside is an HTML page.
~Its primary function is to manipulate and provide access to the respective database as per the request.
~In this any server-side technology can be used and it does not
depend on the client.
~It runs on the webserver.
~The primary advantage is its ability to highly customize, response
requirements, access rights based on user.
~It provides more security for data.
~It is a technique that uses scripts on the webserver to produce a response that is customized for each client's request.
~PHP, Python, Java, Ruby are used.
~It is all about interacting with the servers.
~It surge the processing load on the server.

Q.26.Explain pop-up boxes in JavaScript with example.

ANS:-
JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

~Alert Box:-
An alert box is often used if you want to make sure information comes through to the user.
When an alert box pops up, the user will have to click "OK" to proceed.
Syntax
window.alert("sometext");
The window.alert() method can be written without the window prefix.

Example
alert("I am an alert box!");

~Confirm Box:-
A confirm box is often used if you want the user to verify or accept something.
When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.
Syntax
window.confirm("sometext");
The window.confirm() method can be written without the window prefix.

Example
```
if (confirm("Press a button!"))
{
  txt = "You pressed OK!";
}
else
{
  txt = "You pressed Cancel!";
}
```

~Prompt Box:-
A prompt box is often used if you want the user to input a value before entering a page.
When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.
Syntax
window.prompt("sometext","defaultText");
The window.prompt() method can be written without the window prefix.

Example
```
let person = prompt("Please enter your name", "Harry Potter");
let text;
if (person == null || person == "")
{
  text = "User cancelled the prompt.";
}
else
{
  text = "Hello " + person + "! How are you today?";
}
```

Q.27.Explain DOM in javascript.

ANS:-
The document object represents the whole html document.

When html document is loaded in the browser, it becomes a document object. It is the root element that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page.

As mentioned earlier, it is the object of window.

Methods of document object
We can access and change the contents of document by its methods.

The important methods of document object are as follows:

Method	Description
write("string")	writes the given string on the doucment.
writeln("string")	writes the given string on the doucment with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
getElementsByClassName()	returns all the elements having the given class name.

dom example:-

```
<script type="text/javascript">
function printvalue(){
var name=document.form1.name.value;
alert("Welcome: "+name);
}
</script>

<form name="form1">
Enter Name:<input type="text" name="name"/>
<input type="button" onclick="printvalue()" value="print name"/>
</form>
```

Q.28.Explain Callback function in JavaScript with example.

ANS:-
A function is a block of code that performs a certain task when called. For example,

```
// function
function greet(name) {
   console.log('Hi' + ' ' + name);
}

greet('Peter'); // Hi Peter
```

In the above program, a string value is passed as an argument to the greet() function.

In JavaScript, you can also pass a function as an argument to a function. This function that is passed as an argument inside of another function is called a callback function. For example,

```
// function
function greet(name, callback) {
   console.log('Hi' + ' ' + name);
   callback();
}

// callback function
function callMe() {
```

```
    console.log('I am callback function');
}

// passing function as an argument
greet('Peter', callMe);
```

Output

```
Hi Peter
I am callback function
```

In the above program, there are two functions. While calling the greet() function, two arguments (a string value and a function) are passed.

The callMe() function is a callback function.

Benefit of Callback Function

The benefit of using a callback function is that you can wait for the result of a previous function call and then execute another function call.

In this example, we are going to use the setTimeout() method to mimic the program that takes time to execute, such as data coming from the server.

Example: Program with setTimeout()

```
//  program that shows the delay in execution

function greet() {
    console.log('Hello world');
}

function sayName(name) {
    console.log('Hello' + ' ' + name);
}

// calling the function
setTimeout(greet, 2000);
sayName('John');
```

Run Code

Output

```
Hello John
Hello world
```

As you know, the setTimeout() method executes a block of code after the specified time.

Here, the greet() function is called after 2000 milliseconds (2 seconds). During this wait, the sayName('John'); is executed. That is why Hello John is printed before Hello world.

The above code is executed asynchronously (the second function; sayName() does not wait for the first function; greet() to complete).

Example: Using a Callback Function

In the above example, the second function does not wait for the first function to be complete. However, if you want to wait for the result of the previous function call before the next statement is executed, you can use a callback function. For example,

```
// Callback Function Example
```

```
function greet(name, myFunction) {
   console.log('Hello world');

   // callback function
   // executed only after the greet() is executed
   myFunction(name);
}

// callback function
function sayName(name) {
   console.log('Hello' + ' ' + name);
}

// calling the function after 2 seconds
setTimeout(greet, 2000, 'John', sayName);
```

Run Code

Output

```
Hello world
Hello John
```

Q.29.What is JavaScript event handling? List the major events and show use of at least one event by writing JavaScript code.
OR
Demonstrate the use of onchange, onmouseover, onmouseout, onkeypress, onload, onfocus events with proper example.

ANS:-
The change in the state of an object is known as an Event. In html, there are various events which represents that some activity is performed by the user or by the browser. When javascript code is included in HTML , js react over these events and allow the execution. This process of reacting over the events is called Event Handling. Thus, js handles the HTML events via Event Handlers.

For example, when a user clicks over the browser, add js code, which will execute the task to be performed on the event.

Some of the HTML events and their event handlers are:

Mouse events:

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

Keyboard events:

Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key

Form events:

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

Window/Document events:

Event Performed	Event Handler	Description
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it
resize	onresize	When the visitor resizes the window of the browser

Let's discuss some examples over events and their handlers.

Click Event example:-

```
<html>
<head> Javascript Events </head>
<body>
<script language="Javascript" type="text/Javascript">
   <!--
   function clickevent()
   {
      document.write("This is JavaTpoint");
   }
   //-->
</script>
<form>
<input type="button" onclick="clickevent()" value="Who's this?"/>
</form>
</body>
</html>
```

Q.30.List out JavaScript’s inbuilt Objects. Explain any three JavaScript’s inbuilt Objects with proper example.

ANS:-
1. Math Object
2. Date Object
3. String Object
4. array object
5. boolean object
6. regexp object

1. Math Object
Math object is a built-in static object.
It is used for performing complex math operations.
Math Properties

Math Property	Description

SQRT2	Returns square root of 2.
PI	Returns Π value.
E \	Returns Euler's Constant.
LN2	Returns natural logarithm of 2.
LN10	Returns natural logarithm of 10.
LOG2E	Returns base 2 logarithm of E.
LOG10E	Returns 10 logarithm of E.

Math Methods

Methods	Description
abs()	Returns the absolute value of a number.
acos()	Returns the arccosine (in radians) of a number.
ceil()	Returns the smallest integer greater than or equal to a number.
cos()	Returns cosine of a number.
floor()	Returns the largest integer less than or equal to a number.
log()	Returns the natural logarithm (base E) of a number.
max()	Returns the largest of zero or more numbers.
min()	Returns the smallest of zero or more numbers.
pow()	Returns base to the exponent power, that is base exponent.

Example: Simple Program on Math Object Methods

```
<html>
   <head>
      <title>JavaScript Math Object Methods</title>
   </head>
   <body>
      <script type="text/javascript">

         var value = Math.abs(20);
         document.write("ABS Test Value : " + value +"<br>");

         var value = Math.acos(-1);
         document.write("ACOS Test Value : " + value +"<br>");

         var value = Math.asin(1);
         document.write("ASIN Test Value : " + value +"<br>");

         var value = Math.atan(.5);
         document.write("ATAN Test Value : " + value +"<br>");
      </script>
   </body>
</html>
```

Output

```
ABS Test Value : 20
ACOS Test Value : 3.141592653589793
ASIN Test Value : 1.5707963267948966
ATAN Test Value : 0.4636476090008061
```

Example: Simple Program on Math Object Properties

```
<html>
```

```
    <head>
        <title>JavaScript Math Object Properties</title>
    </head>
    <body>
        <script type="text/javascript">
            var value1 = Math.E
            document.write("E Value is :" + value1 + "<br>");

            var value2 = Math.LN2
            document.write("LN2 Value is :" + value2 + "<br>");

            var value3 = Math.LN10
            document.write("LN10 Value is :" + value3 + "<br>");

            var value4 = Math.PI
            document.write("PI Value is :" + value4 + "<br>");
        </script>
    </body>
</html>
```

Output:

E Value is :2.718281828459045
LN2 Value is :0.6931471805599453
LN10 Value is :2.302585092994046
PI Value is :3.141592653589793

2. Date Object

Date is a data type.
Date object manipulates date and time.
Date() constructor takes no arguments.
Date object allows you to get and set the year, month, day, hour, minute, second and millisecond fields.
Syntax:
var variable_name = new Date();

Example:
var current_date = new Date();

Date Methods

Methods	Description
Date()	Returns current date and time.
getDate()	Returns the day of the month.
getDay()	Returns the day of the week.
getFullYear()	Returns the year.
getHours()	Returns the hour.
getMinutes()	Returns the minutes.
getSeconds()	Returns the seconds.
getMilliseconds()	Returns the milliseconds.
getTime()	Returns the number of milliseconds since January 1, 1970 at 12:00 AM.
getTimezoneOffset()	Returns the timezone offset in minutes for the current locale.
getMonth()	Returns the month.
setDate()	Sets the day of the month.
setFullYear()	Sets the full year.
setHours()	Sets the hours.

setMinutes()	Sets the minutes.
setSeconds()	Sets the seconds.
setMilliseconds()	Sets the milliseconds.
setTime()	Sets the number of milliseconds since January 1, 1970 at 12:00 AM.
setMonth()	Sets the month.
toDateString()	Returns the date portion of the Date as a human-readable string.
toLocaleString()	Returns the Date object as a string.
toGMTString()	Returns the Date object as a string in GMT timezone.
valueOf()	Returns the primitive value of a Date object.

Example : JavaScript Date() Methods Program

```
<html>
   <body>
   <center>
      <h2>Date Methods</h2>
      <script type="text/javascript">
         var d = new Date();
         document.write("<b>Locale String:</b> " + d.toLocaleString()+"<br>");
         document.write("<b>Hours:</b> " + d.getHours()+"<br>");


         document.write("<b>Day:</b> " + d.getDay()+"<br>");


         document.write("<b>Month:</b> " + d.getMonth()+"<br>");
         document.write("<b>FullYear:</b> " + d.getFullYear()+"<br>");
         document.write("<b>Minutes:</b> " + d.getMinutes()+"<br>");
      </script>
   </center>
   </body>
</html>
```

Output:
date object

3. String Object

String objects are used to work with text.
It works with a series of characters.
Syntax:
var variable_name = new String(string);

Example:
var s = new String(string);

String Properties

Properties	Description
length	It returns the length of the string.
prototype	It allows you to add properties and methods to an object.
constructor	It returns the reference to the String function that created the object.

String Methods

Methods	Description
charAt()	It returns the character at the specified index.

charCodeAt()	It returns the ASCII code of the character at the specified position.
concat()	It combines the text of two strings and returns a new string.
indexOf()	It returns the index within the calling String object.
match()	It is used to match a regular expression against a string.
replace()	It is used to replace the matched substring with a new substring.
search()	It executes the search for a match between a regular expression.
slice()	It extracts a session of a string and returns a new string.
split()	It splits a string object into an array of strings by separating the string into the substrings.
toLowerCase()	It returns the calling string value converted lower case.
toUpperCase()	Returns the calling string value converted to uppercase.

Example : JavaScript String() Methods Program

```
<html>
   <body>
   <center>
      <script type="text/javascript">
         var str = "CareerRide Info";
         var s = str.split();
         document.write("<b>Char At:</b> " + str.charAt(1)+"<br>");
         document.write("<b>CharCode At:</b> " + str.charCodeAt(2)+"<br>");
         document.write("<b>Index of:</b> " + str.indexOf("ide")+"<br>");
         document.write("<b>Lower Case:</b> " + str.toLowerCase()+"<br>");
         document.write("<b>Upper Case:</b> " + str.toUpperCase()+"<br>");
      </script>
   <center>
   </body>
</html>
```

Output:
string object

Q.31.How user defined Objects are created in Javascript? explain with proper example.

ANS:-
We can define a custom object type by writing a constructor function where the name of the function should start with an uppercase alphabet for example Car, Cup, Human, etc.

The constructor function is defined just as we define any other user-defined JavaScript Function.

Here is the syntax for defining a constructor function:

```
function Xyz(param1, param2, ... , paramN)
{
   this.param1 = param1;
   this.param2 = param2;
   ...
   this.paramN = paramN;
}
```

Once we have the constructor function defined, we can use it to create an object using the new keyword as follows:

```
let customObj = new Xyz(paramValue1, paramValue2, paramValue3, ... , paramValueN);
```

Using this Keyword

In JavaScript we use this keyword to reference the current object, hence in the constructor function we create new properties (param1, param2, ...) for the object type and we set the values for the parameters provided at the time of object creation using the new keyword.

User-defined objects
Objects that you have defined using custom constructor functions. Let's see an example below,

```
//you define your constructor function
function Employee(name, id) {
  this.name = name;
  this.id = id;
}
//extend the prototye
Employee.prototype.getName = function() {
  return this.name;
}
Employee.prototype.setID = function(ID) {
  this.id = ID;
}
//now start creating employee objects
var e1 = new Employee("John", 1234);
var e2 = new Employee("Lenon", 3481);
//and so on
//now access the properties from the prototype
e1.getName(); //"John"
e2.setID(8888);
```

So your e1, e2 instances or objects can read properties from the prototype of the Employee constructor function. This is an example of prototypal inheritance.

Properties and methods added to the prototype of the constructor function can be seen by instances created from the constructor function.

But if you notice, you will have some pre-defined methods and properties in your custom created object as well.

```
e1.constructor; //which is a reference to Employee function e1.toString(); //"[object Object]"
e1.valueOf(); //Employee {name: "John", id: 1234}
```

Q.32.Explain concept of Array in JavaScript. Also Demonstrate use of any three methods of Array with proper example.

ANS:-
JavaScript array is an object that represents a collection of similar type of elements.

There are 3 ways to construct array in JavaScript

By array literal
By creating instance of Array directly (using new keyword)
By using an Array constructor (using new keyword)
1) JavaScript array literal
The syntax of creating array using array literal is given below:

```
var arrayname=[value1,value2.....valueN];
```

As you can see, values are contained inside [ ] and separated by , (comma).
Play Videox

Methods	Description
concat()	It returns a new array object that contains two or more merged arrays.
indexOf()	It searches the specified element in the given array and returns the index of the first match.
isArray()	It tests if the passed value ia an array.
join()	It joins the elements of an array as a string.
keys()	It creates an iterator object that contains only the keys of the array, then loops through these keys.
lastIndexOf()	It searches the specified element in the given array and returns the index of the last match.
map()	It calls the specified function for every array element and returns the new array
of()	It creates a new array from a variable number of arguments, holding any type of argument.
pop()	It removes and returns the last element of an array.
push()	It adds one or more elements to the end of an array.
reverse()	It reverses the elements of given array.
reduce	It executes a provided function for each value from left to right and reduces the array to a single value.
reduceRight()	It executes a provided function for each value from right to left and reduces the array to a single value.
some()	It determines if any element of the array passes the test of the implemented function.
shift()	It removes and returns the first element of an array.
slice()	It returns a new array containing the copy of the part of the given array.
sort()	It returns the element of the given array in a sorted order.
splice()	It add/remove elements to/from the given array.
toString()	It converts the elements of a specified array into string form, without affecting the original array.
unshift()	It adds one or more elements in the beginning of the given array.

~Converting Arrays to Strings
The JavaScript method toString() converts an array to a string of (comma separated) array values.

Example
```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.toString();
```
Result:

Banana,Orange,Apple,Mango

~The join() method also joins all array elements into a string.

It behaves just like toString(), but in addition you can specify the separator:

Example
```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.join(" * ");
```
Result:

Banana * Orange * Apple * Mango

~JavaScript Array pop()
The pop() method removes the last element from an array:

Example
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.pop();

~JavaScript Array push()
The push() method adds a new element to an array (at the end):

Example
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.push("Kiwi");

~JavaScript Array shift()
The shift() method removes the first array element and "shifts" all other elements to a lower index.

Example
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.shift();

~JavaScript Array unshift()
The unshift() method adds a new element to an array (at the beginning), and "unshifts" older elements:

Example
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.unshift("Lemon");

~Merging (Concatenating) Arrays
The concat() method creates a new array by merging (concatenating) existing arrays:

Example (Merging Two Arrays)
const myGirls = ["Cecilie", "Lone"];
const myBoys = ["Emil", "Tobias", "Linus"];

const myChildren = myGirls.concat(myBoys);

~JavaScript Array splice()
The splice() method can be used to add 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 splice() method returns an array with the deleted items:\

~Using splice() to Remove Elements
With clever parameter setting, you can use splice() to remove elements without leaving "holes" in the array:

Example

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(0, 1);
```
The first parameter (0) defines the position where new elements should be added (spliced in).

The second parameter (1) defines how many elements should be removed.

The rest of the parameters are omitted. No new elements will be added.

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

This example slices out a part of an array starting from array element 1 ("Orange"):

Example
```
const fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
const citrus = fruits.slice(1);
```

Q.33.Write short note on JSON.

ANS:-
JSON or JavaScript Object Notation is a lightweight text-based open standard designed for human-readable data interchange. Conventions used by JSON are known to programmers, which include C, C++, Java, Python, Perl, etc.

JSON stands for JavaScript Object Notation.

The format was specified by Douglas Crockford.

It was designed for human-readable data interchange.

It has been extended from the JavaScript scripting language.

The filename extension is .json.

JSON Internet Media type is application/json.

The Uniform Type Identifier is public.json.

~Uses of JSON
It is used while writing JavaScript based applications that includes browser extensions and websites.

JSON format is used for serializing and transmitting structured data over network connection.

It is primarily used to transmit data between a server and web applications.

Web services and APIs use JSON format to provide public data.

It can be used with modern programming languages.

~Characteristics of JSON
JSON is easy to read and write.
It is a lightweight text-based interchange format.
JSON is language independent.
Simple Example in JSON
The following example shows how to use JSON to store information related to books based on their topic and edition.

```
{
  "book": [

    {
      "id":"01",
      "language": "Java",
      "edition": "third",
      "author": "Herbert Schildt"
    },

    {
      "id":"07",
      "language": "C++",
      "edition": "second",
      "author": "E.Balagurusamy"
    }
  ]
}
```

Q.34.Demonstrate JavaScript Form Validation with proper example.

ANS:-
JavaScript Form Validation
JavaScript form validation
Example of JavaScript validation
JavaScript email validation
It is important to validate the form submitted by the user because it can have inappropriate values. So, validation is must to authenticate user.

JavaScript provides facility to validate the form on the client-side so data processing will be faster than server-side validation. Most of the web developers prefer JavaScript form validation.

Through JavaScript, we can validate name, password, email, date, mobile numbers and more fields.

JavaScript Form Validation Example
In this example, we are going to validate the name and password. The name can’t be empty and password can’t be less than 6 characters long.
Play Videox

Here, we are validating the form on form submit. The user will not be forwarded to the next page until given values are correct.

```
<script>
function validateform(){
var name=document.myform.name.value;
var password=document.myform.password.value;

if (name==null || name==""){
  alert("Name can't be blank");
  return false;
}else if(password.length<6){
  alert("Password must be at least 6 characters long.");
  return false;
  }
}
```

```
</script>
<body>
<form name="myform" method="post" action="abc.jsp" onsubmit="return validateform()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
```

Q.35.Write a Java Script code to display Fibonacci Series of given number. Number should be entered by user through text box.

ANS:-

```
<html>
<head>
<title> Fibonacci Series in JavaScript </title>
</head>
<body>
<script>
// declaration of the variables
var n1 = 0,  n2 = 1, next_num, i;
var num = parseInt (prompt (" Enter the limit for Fibonacci Series "));
document.write( "Fibonacci Series: ");
for ( i = 1; i <= num; i++)
{  document.write (" <br> " +  n1); // print the n1
   next_num = n1 + n2; // sum of n1 and n2 into the next_num

   n1 = n2; // assign the n2 value into n2
   n2 = next_num; // assign the next_num into n2
}

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

Q.36.Write a JavaScript code to change the background color of page at specific time interval.

ANS:-

```
<!DOCTYPE HTML>
<html>
        <head>
                <title>
                        How to change the background color
                        after clicking the button ?
                </title>
        </head>

        <body style = "text-align:center;">

                <h1 style = "color:green;" >
                        hello everyone
                </h1>

                <p id = "GFG_UP" style =
                        "font-size: 16px; font-weight: bold;">
                </p>
```

```
		<button onclick = "gfg_Run()">
			Click here
		</button>

		<p id = "GFG_DOWN" style =
			"color:green; font-size: 20px; font-weight: bold;">
		</p>

		<script>
			var el_up = document.getElementById("GFG_UP");
			var el_down = document.getElementById("GFG_DOWN");
			var str = "Click on button to change the background color";

			el_up.innerHTML = str;

			function changeColor(color) {
				document.body.style.background = color;
			}


			function gfg_Run() {
				changeColor('yellow');
				el_down.innerHTML = "Background Color changed";
			}
		</script>
	</body>
</html>
```

Q.37.Write a JavaScript code which takes integer number as input and Tells whether the number is prime or not.

ANS:-

```
<!DOCTYPE html>
<html>

<head>
	<title>
		Check a number is Prime or
		not using JavaScript
	</title>

	<script type="text/javascript">

		// Function to check prime number
		function p() {

			var n, i, flag = true;

			// Getting the value form text
			// field using DOM
			n = document.myform.n.value;
			n = parseInt(n)
			for(i = 2; i <= n - 1; i++)
				if (n % i == 0) {
					flag = false;
					break;
```

```
                        }

                        // Check and display alert message
                    if (flag == true)
                        alert(n + " is prime");
                    else
                        alert(n + " is not prime");
                }
        </script>
</head>

<body>
        <center>
                <h1>GeeksforGeeks</h1>

                <h4>check number is prime or not</h4>

                <hr color="Green">

                <form name="myform">
                        Enter the number:
                        <input type="text" name=n value="">

                        <br><br>

                        <input type="button" value="Check" onClick="p()">
                        <br>
                </form>
        </center>
</body>

</html>
```

Q.38.Write JavaScript code that display the text “SILVER OAK UNIVERSITY” with increasing font size in interval of 1 second in blue color. When font size reaches to 50px it should stop

ANS:-

```
<body>

<p id="p1" style="color: blue"> silver oak University </p>

</body>

<script>

var P1 = document.getElementById("P1");

let i=16;

var a=setInterval(()=> {

P1.style font size=i+ "px";
i = i+4;
if (i>=50){
 clearInterval(a);
```

```
}
},1000)

</script>
```

unit 7

Q.56. What are the advantages of Node JS? Where to use Node JS?

AnS:-

High-performance for Real-time Applications
Easy Scalability for Modern Applications
Cost-effective with Fullstack JS
Community Support to Simplify Development
Easy to Learn and Quick to Adapt
Helps in building Cross-functional Teams
Improves App Response Time and Boosts Performance
Reduces Time-to-Market of your applications
Extensibility to Meet Customized Requirements
Reduces Loading Time by Quick Caching
Helps in Building Cross-Platform Applications

Node. js is primarily used for non-blocking, event-driven servers, due to its single-threaded nature. It's used for traditional web sites and back-end API services, but was designed with real-time, push-based architectures in mind.

Node. js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node. js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Q.57.Explain the features of Node JS.

ANS:-
1.Asynchronous and Event-Driven: The Node.js library's APIs are all asynchronous (non-blocking) in nature. A server built with Node.JS never waits for data from an API. from an API. After accessing an API, the server moves on to the next one. In order to receive and track responses of previous API requests, it uses a notification mechanism called Events.

2.Single-Threaded: Node.js employs a single-threaded architecture with event looping, making it very scalable. In contrast to typical servers, which create limited threads to process requests, the event mechanism allows the node.js server to reply in a non-blocking manner and makes it more scalable. When compared to traditional servers like Apache HTTP Server, Node.js uses a single-threaded program that can handle a considerably larger number of requests.

3.Scalable: NodeJs addresses one of the most pressing concerns in software development: scalability. Nowadays, most organizations demand scalable software. NodeJs can also handle concurrent requests efficiently. It has a cluster module that manages load balancing for all CPU cores that are active. The capability of NodeJs to partition applications horizontally is its most appealing feature. It achieves this through the use of child processes. This allows the organizations to provide distinct app versions to different target audiences, allowing them to cater to client preferences for customization.

4.Quick execution of code: Node.js makes use of the V8 JavaScript Runtime motor, which is also used by Google Chrome. Hub provides a wrapper for the JavaScript motor, which makes the runtime motor faster. As a result, the preparation of requests inside Node.js becomes faster as well.
Cross-platform compatibility: NodeJS may be used on a variety of systems, including Windows, Unix, Linux, Mac OS X, and mobile devices. It can be paired with the appropriate package to generate a self-sufficient executable.

5.Uses JavaScript: JavaScript is used by the Node.js library, which is another important aspect of Node.js from the engineer's perspective. Most of the engineers are already familiar with JavaScript. As a result, a designer who is familiar with JavaScript will find that working with Node.js is much easier.

6.Fast data streaming: When data is transmitted in multiple streams, processing them takes a long time. Node.js processes data at a very fast rate. It processes and uploads a file simultaneously, thereby saving a lot of time. As a result, NodeJs improves the overall speed of data and video streaming.

7.No Buffering: In a Node.js application, data is never buffered.

Q.58.What is NPM? Explain the use of Package Manager in Node.js with suitable example.

ANS:-
NPM (Node Package Manager) is the default package manager for Node.js and is written entirely in Javascript. Developed by Isaac Z. Schlueter, it was initially released in January 12, 2010. NPM manages all the packages and modules for Node.js and consists of command line client npm. It gets installed into the system with installation of Node.js. The required packages and modules in Node project are installed using NPM.
A package contains all the files needed for a module and modules are the JavaScript libraries that can be included in Node project according to the requirement of the project.
NPM can install all the dependencies of a project through the package.json file. It can also update and uninstall packages. In the package.json file, each dependency can specify a range of valid versions using the semantic versioning scheme, allowing developers to auto-update their packages while at the same time avoiding unwanted breaking changes.

Node Package Manager provides two main functionalities:

It provides online repositories for node.js packages/modules which are searchable on search.nodejs.org
It also provides command line utility to install Node.js packages, do version management and dependency management of Node.js packages.

Node Package Manager (NPM) is an open-source software library that has over 800,000 code packages. In simple terms, we can say that NPM is a command-line tool that installs, updates, or uninstalls node. js packages of an application.

Q.59.Explain REPL terminal in Node.js .

ANS:-
The term REPL stands for Read Eval Print and Loop. It specifies a computer environment like a window console or a Unix/Linux shell where you can enter the commands and the system responds with an output in an interactive mode.

REPL Environment
The Node.js or node come bundled with REPL environment. Each part of the REPL environment has a specific work.

Read: It reads user's input; parse the input into JavaScript data-structure and stores in memory.

Eval: It takes and evaluates the data structure.

Print: It prints the result.

Loop: It loops the above command until user press ctrl-c twice.

How to start REPL
You can start REPL by simply running "node" on the command prompt. See this:

node.js repl 1
You can execute various mathematical operations on REPL Node.js command prompt:

Node.js Simple expressions
After starting REPL node command prompt put any mathematical expression:

Example: >10+20-5
25
node.js repl 2
Example2: >10+12 + (5*4)/7
node.js repl 3
Using variable
Variables are used to store values and print later. If you don't use var keyword then value is stored in the variable and printed whereas if var keyword is used then value is stored but not printed. You can print variables using console.log().

Example:

node.js repl 4
Node.js Multiline expressions
Node REPL supports multiline expressions like JavaScript. See the following do-while loop example:

```
var x = 0
undefined
> do {
... x++;
... console.log("x: " + x);
... } while ( x < 10 );
```

node.js repl 5
Node.js Underscore Variable
You can also use underscore _ to get the last result.

Example:

node.js repl 6
Node.js REPL Commands

Commands	Description
ctrl + c	It is used to terminate the current command.
ctrl + c twice	It terminates the node repl.
ctrl + d	It terminates the node repl.
up/down keys	It is used to see command history and modify previous commands.
tab keys	It specifies the list of current command.
.help	It specifies the list of all commands.
.break	It is used to exit from multi-line expressions.

.clear It is used to exit from multi-line expressions.
.save filename It saves current node repl session to a file.
.load filename It is used to load file content in current node repl session.

Q.60.Explain Rendering HTML in Node js.

ANS:-
In this article we will cover on how to implement how to render html file in node js express. i explained simply step by step node js route to html page. you can understand a concept of node js redirect to html page. you will learn node js express render html.

In this example, i will give you very simple example of how to render html file in node js express project. we will use sendFile() to render html file, let's see bellow example.

Create Node Project:

you can run bellow command to create node app.

```
mkdir my-request-app
cd my-request-app
```

```
npm init
```
Install Express:

```
npm install express
```
Create Server.js File

let's create server.js file with get and post route.

server.js

```
var express = require('express');
var app = express();

app.get('/',function(req,res) {
  res.sendFile(__dirname + '/index.html');
});

app.listen(3000, () => console.log(`App listening on port 3000`))
```
Create index.html File

index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title></title>
</head>
<body>

<p>This is simple html file call by ItSolutionStuff.com!</p>

</body>
```

```
</html>
```
Output:

This is simple html file call by ItSolutionStuff.com!

Q.61.Explain Rendering JSON data in Node js.

ANS:-
When you want to store data between server restarts with Node, JSON files are a simple and convenient choice. Whether you are reading a config file or persisting data for your application, Node has some built in utilities that make it easy to read and write JSON files. Using JSON files in your app can be a useful way to persist data. We will look at a few different methods for working with JSON files.

Read JSON data from disk
Learn to use fs module to interact with the filesystem
Persist data to a JSON file
Use JSON.parse and JSON.stringify to convert data to and from JSON format

Read JSON From File System In NodeJS:
At first I tried googling about it, I found a solution, which shows example with file system support of nodejs(fs module). But, I don't really see any meaning of that at all, as we can simply do the same thing by:

```
var jsonObj = require("./path/to/myjsonfile.json");
```
Here, NodeJS automatically read the file, parse the content to a JSON object and assigns that to the left hand side variable. It's as simple as that!

Q.62.Write a Node JS code to create a server.

ANS:-
```
var http = require('http'); // 1 - Import Node.js core module

var server = http.createServer(function (req, res) {   // 2 - creating server

   //handle incomming requests here..

});

server.listen(5000); //3 - listen for any incoming requests

console.log('Node.js web server at port 5000 is running..')
```

output:-
```
C:\> node server.js
Node.js web server at port 5000 is running..
```

Q.63.Write a code for reading a file asynchronously (non-blocking code) in Node.js

ANS:-
```
var fs = require('fs');

fs.readFile('DATA', 'utf8', function(err, contents) {
   console.log(contents);
});

console.log('after calling readFile');
```

Q.64.Explain concept of routing in Node JS with example

ANS:-
Routing defines the way in which the client requests are handled by the application endpoints.

Implementation of routing in Node.js: There are two ways to implement routing in node.js which are listed below:

By Using Framework
Without using Framework
Using Framework: Node has many frameworks to help you to get your server up and running. The most popular is Express.js.
Routing with Express in Node: Express.js has an “app” object corresponding to HTTP. We define the routes by using the methods of this “app” object. This app object specifies a callback function, which is called when a request is received. We have different methods in app object for a different type of request.

For GET request use app.get() method:

```
var express = require('express')
var app = express()

app.get('/', function(req, res) {
   res.send('Hello Sir')
})
```

For POST request use app.post() method:

```
var express = require('express')
var app = express()

app.post('/', function(req, res) {
   res.send('Hello Sir')
})
```

For handling all HTTP methods (i.e. GET, POST, PUT, DELETE etc.) use app.all() method:

```
var express = require('express')
var app = express()

app.all('/', function(req, res) {
   console.log('Hello Sir')
   next()   // Pass the control to the next handler
})
```

The next() is used to hand off the control to the next callback. Sometimes we use app.use() to specify the middleware function as the callback.

So, to perform routing with the Express.js you have only to load the express and then use the app object to handle the callbacks according to the requirement.

Routing without Framework: Using the frameworks is good to save time, but sometimes this may not suit the situation. So, a developer may need to build up their own server without other dependencies.