Chapter 1

Ans 1

Architecture of Web Browser

The architecture of a web browser has certain components. They are Controller/Dispatcher, Interpreter, and Client Programs. The control unit in a CPU is a **controller** that takes in the input, interprets it, and then instructs the device to work in a certain way.
The **interpreter** receives this info from the controller to get ahead in the given task step by step. Lastly, the **client program** has sets of protocols to complete a particular service. Some common names are – HTTP, SMTP, FTP, etc.

Websites

As mentioned above, the location on a web browser using a URL opens up a website. They are of two types –
1. Static Websites – These are flat or stationary websites made by the client that only they can change. The users can read and watch them but cannot interact with the content in any way. They use HTML language to create as the information is fixed.
2. Dynamic Websites – These websites keep updating and the users will find new information every time. The ajax technology allows users to change information as they want to make it very dynamic.

Web Pages

The web pages are the places where users find all sorts of information. The hyperlinks to many other pages allowing users to switch and explore. And a collection of these pages make up a website. Web pages are also of two types like websites – Static and Dynamic. The brief for them is the same as above.

Under dynamic web pages, they can be either Server-side dynamic web pages or Client-side dynamic web pages. The server-side dynamic page has server-side scripting while the other one has client-side scripting using the Document Object Model. The scripting has become quite common nowadays and some of them are –

Client-Side Web Scripting

1. JavaScript – prototype-based scripting that inherits its name from java and is saved as a .js extension.

2. ActionScript – an object-oriented language used to target adobe flash player for website development.
3. Dart – source to source compiler and is the open-source language by Google.
4. VBScript – open source web language by Microsoft that has static typing class-based object programming.

Server-side Web Scripting

1. Active Server Pages – supports Component Object Model that uses functions of DLL.
2. ActiveVFP – uses native Visual Foxpro language and database for website creation
3. ASP.net – develops websites, web applications, and web services
4. Java – Java code with byte code creates the website
5. Python – object-oriented and functional programming for website creation
6. WebDNA – uses an embedded database system

Ans 2

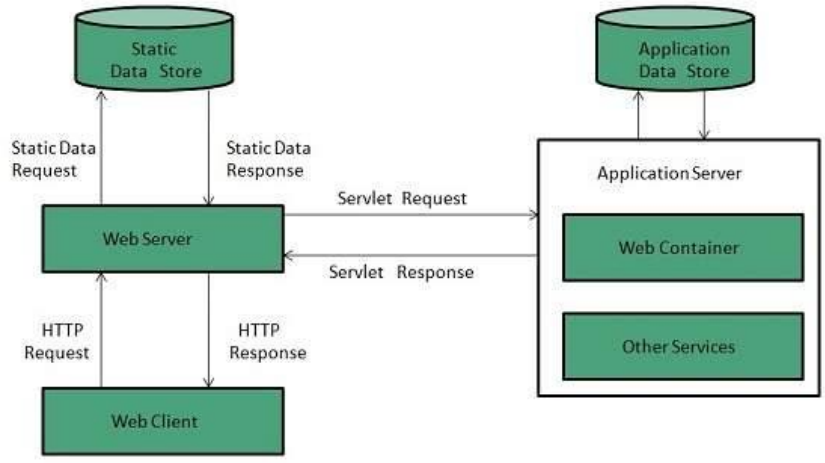
Web server is a computer where the web content is stored. Basically web server is used to host the web sites but there exists other web servers also such as gaming, storage, FTP, email etc.

Web site is collection of web pages whileweb server is a software that respond to the request for web resources.

Web Server Working

Web server respond to the client request in either of the following two ways:

- Sending the file to the client associated with the requested URL.
- Generating response by invoking a script and communicating with database



Key Points

- When client sends request for a web page, the web server search for the requested page if requested page is found then it will send it to client with an HTTP response.
- If the requested web page is not found, web server will the send an **HTTP response:Error 404 Not found.**
- If client has requested for some other resources then the web server will contact to the application server and data store to construct the HTTP response.

Architecture

Web Server Architecture follows the following two approaches:

1. Concurrent Approach
2. Single-Process-Event-Driven Approach.

Concurrent Approach

Concurrent approach allows the web server to handle multiple client requests at the same time. It can be achieved by following methods:

- Multi-process
- Multi-threaded
- Hybrid method.

Multi-processing

In this a single process (parent process) initiates several single-threaded child processes and distribute incoming requests to these child processes. Each of the child processes are responsible for handling single request.

It is the responsibility of parent process to monitor the load and decide if processes should be killed or forked.

Multi-threaded

Unlike Multi-process, it creates multiple single-threaded process.

Hybrid

It is combination of above two approaches. In this approach multiple process are created and each process initiates multiple threads. Each of the threads handles one connection. Using multiple threads in single process results in less load on system resources.

Ans 4

World Wide Web Communication

The World Wide Web is about communication between web **clients** and web **servers**.

Clients are often browsers (Chrome, Edge, Safari), but they can be any type of program or device.

Servers are most often computers in the cloud.

HTTP Request / Response

Communication between clients and servers is done
by **requests** and **responses**:

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

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.

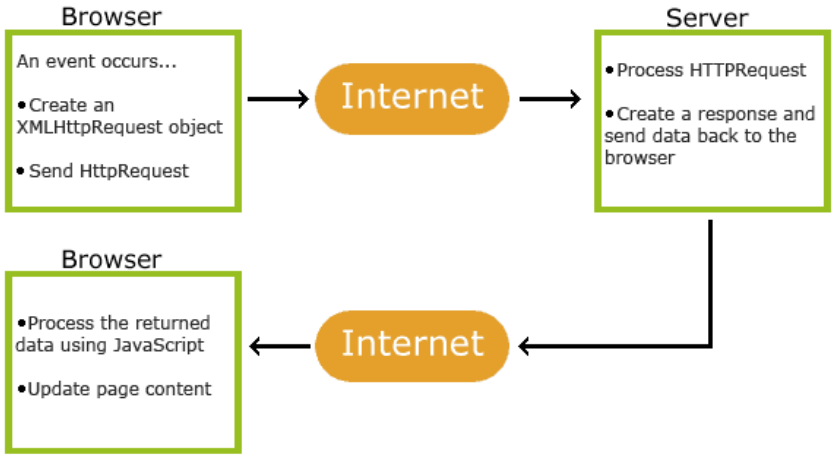
XHR is often used to request and receive data for the purpose of modifying a web page.

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.

The XHR Object is a **Web Developers Dream**, because you can:

- Update a web page without reloading the page
- Request data from a server - after the page has loaded
- Receive data from a server - after the page has loaded
- Send data to a server - in the background

The XHR Object is the underlying concept of **AJAX** and **JSON**:



HTTP Methods

- **GET**
- **POST**
- **PUT**
- **HEAD**
- **DELETE**
- **PATCH**
- **OPTIONS**
- **CONNECT**
- **TRACE**

Compare GET vs. POST

The following table compares the two HTTP methods: GET and POST.

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data

History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

The PUT Method

PUT is used to send data to a server to create/update a resource.

The difference between POST and PUT is that PUT requests are idempotent. That is, calling the same PUT request multiple times will always produce the same result. In contrast, calling a POST request repeatedly have side effects of creating the same resource multiple times.

The HEAD Method

HEAD is almost identical to GET, but without the response body.

In other words, if GET /users returns a list of users, then HEAD /users will make the same request but will not return the list of users.

HEAD requests are useful for checking what a GET request will return before actually making a GET request - like before downloading a large file or response body.

The DELETE Method

The DELETE method deletes the specified resource.

The PATCH Method

The PATCH method is used to apply partial modifications to a resource.

The OPTIONS Method

The OPTIONS method describes the communication options for the target resource.

The CONNECT Method

The CONNECT method is used to start a two-way communications (a tunnel) with the requested resource.

The TRACE Method

The TRACE method is used to perform a message loop-back test that tests the path for the target resource (useful for debugging purposes)

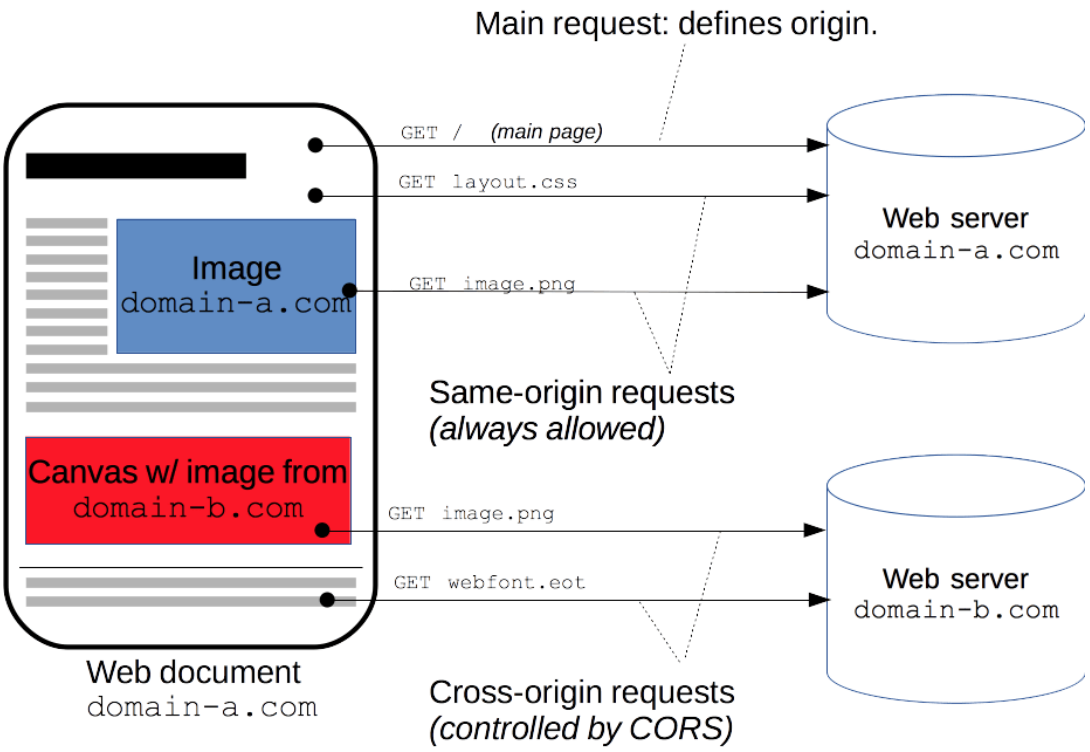
Ans 4

The full form of CORS is **Cross-Origin Resource Sharing** (CORS)

Cross-Origin Resource Sharing (CORS) is an HTTP-header based mechanism that allows a server to indicate any origins (domain, scheme, or port) other than its own from which a browser should permit loading resources. CORS also relies on a mechanism by which browsers make a "preflight" request to the server hosting the cross-origin resource, in order to check that the server will permit the actual request. In that preflight, the browser sends headers that indicate the HTTP method and headers that will be used in the actual request.

An example of a cross-origin request: the front-end JavaScript code served from `https://domain-a.com` uses `XMLHttpRequest` to make a request for `https://domain-b.com/data.json`.

For security reasons, browsers restrict cross-origin HTTP requests initiated from scripts. For example, `XMLHttpRequest` and the Fetch API follow the same-origin policy. This means that a web application using those APIs can only request resources from the same origin the application was loaded from unless the response from other origins includes the right CORS headers.



The CORS mechanism supports secure cross-origin requests and data transfers between browsers and servers. Modern browsers use CORS in APIs such as `XMLHttpRequest` or Fetch to mitigate the risks of cross-origin HTTP requests.

Ans 5

1. The **World Wide Web** (**WWW**), commonly known as the **Web**, is an information system enabling documents and other web resources to be accessed over the Internet.
2. A website **is a collection of publicly accessible, interlinked Web pages that share a single domain name**.
3. **Extensible Markup Language** (**XML**) is a markup language and file format for storing, transmitting, and reconstructing arbitrary data. It defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

Ans 6

Web security is a broad category of security solutions that protect your users, devices, and wider network against internet-based cyberattacks—malware, phishing, and more—that can lead to breaches and data loss. Many web security solutions reduce the security risk to your organization when your users accidentally access malicious files and websites.

In the traditional web security model, a “stack” of security appliances at an internet gateway inspects traffic that passes through it, including some combination of firewall inspection, intrusion prevention system (IPS) scanning, sandboxing, URL filtering, and

various other security and access controls.

Web Security vs. Web Application Security

Before we continue, let's clarify these terms. Web security can also include web application security (also called website security), a subcategory of practices and tools that help secure public-facing websites. For the purposes of this article, we'll be focusing specifically on web security solutions that secure servers or user endpoints as well as the traffic that moves between those and the internet.

What's the Purpose of Web Security?

The massive importance of the internet for modern enterprises—and the accompanying growth in the sophistication, frequency, and impact of cyberattacks—has made web security critical to business continuity. It's your first line of defense against threats that can lead to the exposure of sensitive data, costly ransoms, reputational harm, compliance violations, and a host of other consequences.

Once the domain of mostly small-time hackers, internet-borne threats have evolved into a massive black market business that touches the worlds of organized crime as well as state-sponsored espionage and sabotage. Some of the latest threats are incredibly sophisticated, able to easily fool the untrained eye or bypass legacy security. Plus, with an array of ready-made tools, exploit kits, JavaScript modules, and even fully developed campaigns for sale, even a novice actor can easily launch an attack.

Cybersecurity Ventures estimates that, by 2025, global cybercrime will cost cost US$10.5 trillion annually—a greater profit than the entire world's major illicit drug trade—and half the world's data will live in the cloud. Given what's at stake, it's easy to see why effective web security is so important today.

What are the Benefits of Web Security?

For a modern enterprise, effective web security has broad technical and human benefits:

- **Protect your business and stay compliant** by preventing loss of sensitive data
- **Protect customers and employees** by securing their private information
- **Avoid costly service interruptions** by preventing infections and exploits
- **Offer a better user experience** by helping your users stay safe and productive
- **Maintain customer loyalty and trust** by staying secure and out of the news

Great strides in cloud and mobility technology let your employees and customers connect with you with unprecedented ease and flexibility. Unfortunately, that cuts both

ways, and attackers have more ways to approach your organization's wider attack surface. With the right web protections in place, you can spend more time taking advantage of the benefits, and less worrying about security threats.

What Does Web Security Protect Against?

Web security casts a wide net to protect users and endpoints from malicious emails, encrypted threats, malicious or compromised websites and databases, malicious redirects, hijacking, and more. Let's look at a few of the most common threats in more detail:

- **Ransomware:** These attacks encrypt data, and then demand a ransom payment in exchange for a decryption key. In a double-extortion attack, your data is also exfiltrated.
- **General malware:** Countless variants of malware exist that can lead to anything from data leaks, spying, and unauthorized access to lockouts, errors, and system crashes.
- **Phishing:** Often carried out through email, text messages, or malicious websites, these attacks trick users into things like divulging login credentials or downloading spyware.
- **SQL injection:** These attacks exploit an input vulnerability in a database server, allowing an attacker to execute commands that let them retrieve, manipulate, or delete data.
- **Denial of service (DoS):** These attacks slow or even shut down a network device such as a server by sending it more data than it can process. In distributed DoS—that is, a DDoS attack—this is carried out by many hijacked devices at once.
- **Cross-site scripting (XSS):** In this type of injection attack, an attacker introduces malicious code to a trusted website by entering it in an unprotected user input field.

Ans 7

HTTP GET	HTTP POST
1. In GET method we can not send large amount of data rather limited data is sent because the request parameter is appended into the URL.	In POST method large amount of data can be sent because the request parameter is appended into the body.
2. GET request is comparatively better than Post so it is used more than thePost request.	POST request is comparatively less better than Get so it is used less than the Get request.

3. GET request is comparatively less secure because the data is exposed in the URL bar.	POST request is comparatively more secure because the data is not exposed in the URL bar.
4. Request made through GET method are stored in Browser history.	Request made through POST method is not stored in Browser history.
5. GET method request can be saved as bookmark in browser.	POST method request can not be saved as bookmark in browser.
6. Request made through GET method are stored in cache memory of Browser.	Request made through POST method are not stored in cache memory of Browser.
7. Data passed through GET method can be easily stolen by attackers.	Data passed through POST method can not be easily stolen by attackers.
8. In GET method only ASCII characters are allowed.	In POST method all types of data is allowed.

Chapter 2

Ans 1

An HTML 4 document is composed of three parts:

1. a line containing HTML version information,
2. a declarative header section (delimited by the `HEAD` element),
3. a body, which contains the document's actual content. The body may be implemented by the `BODY` element or the `FRAMESET` element.

White space (spaces, newlines, tabs, and comments) may appear before or after each section. Sections 2 and 3 should be delimited by the `HTML` element.

Here's an example of a simple HTML document:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
   <HEAD>
      <TITLE>My first HTML document</TITLE>
   </HEAD>
   <BODY>
      <P>Hello world!
   </BODY>
</HTML>
```

Ans 2

ordered List or Bulleted List

In HTML unordered list, the list items have no specific order or sequence. An unordered list is also called a Bulleted list, as the items are marked with bullets. It begins with the <ul> tag and and closes with a tag. The list items begin with the <li> tag and end with tag.

Syntax:

```
<ul>List of Items</ul>
```

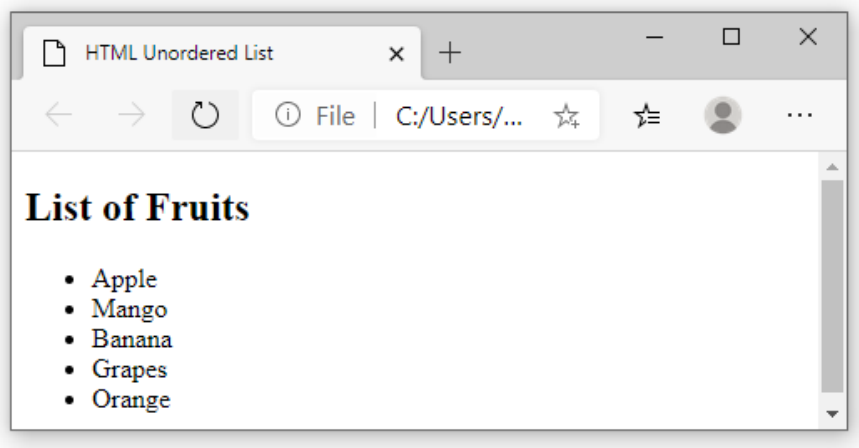
Example of HTML Unordered List

```
<!DOCTYPE html>

<html>
```

```
    <head>
       <title>HTML Unordered List</title>
    </head>
    <body>
    <h2>List of Fruits</h2>
       <ul>
          <li>Apple</li>
          <li>Mango</li>
          <li>Banana</li>
          <li>Grapes</li>
          <li>Orange</li>
       </ul>
    </body>
</html>
```

Output:



Ordered List or Numbered List (ol)

In HTML, all the list items in an ordered list are marked with numbers by default instead of bullets. An HTML ordered list starts with the <ol> tag and ends with the tag. The list items start with the <li> tag and end with tag.

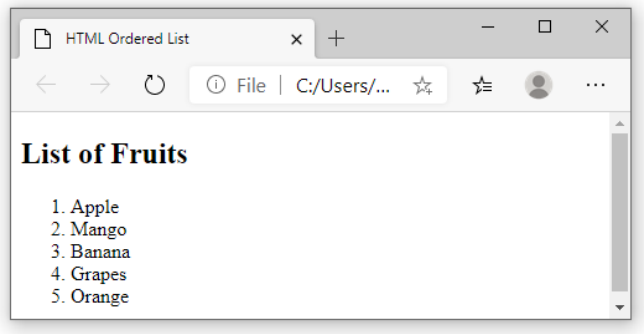
Syntax:

```
<ol>List of Items</ol>
```

Example of HTML Ordered List

```
<!DOCTYPE html>
<html>
   <head>
      <title>HTML Ordered List</title>
   </head>
   <body>
   <h2>List of Fruits</h2>
      <ol>
         <li>Apple</li>
         <li>Mango</li>
         <li>Banana</li>
         <li>Grapes</li>
         <li>Orange</li>
      </ol>
   </body>
</html>
```

Output:



Ans 3

HTML Images Syntax

The HTML `<img>` tag is used to embed an image in a web 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 is empty, it contains attributes only, and does not have a closing tag.

The `<img>` tag has two required attributes:

- src - Specifies the path to the image
- alt - Specifies an alternate text for the image

Syntax

```
<img src="url" alt="alternatetext">
```

The src Attribute

The required `src` attribute specifies the path (URL) to the image.

Note: When a web page loads, it is the browser, at that moment, that gets the image from a web server and inserts it into the page. Therefore, make sure that the image actually stays in the same spot in relation to the web page, otherwise

your visitors will get a broken link icon. The broken link icon and the `alt` text are shown if the browser cannot find the image.

Example

```
<img src="img_chania.jpg" alt="Flowers in Chania">
```

The alt Attribute

The required `alt` attribute provides an alternate text for an image, if the user for some reason cannot view it (because of slow connection, an error in the src attribute, or if the user uses a screen reader).

The value of the `alt` attribute should describe the image:

Example

```
<img src="img_chania.jpg" alt="Flowers in Chania">
```

Try it Yourself »

If a browser cannot find an image, it will display the value of the `alt` attribute:

Example

```
<img src="wrongname.gif" alt="Flowers in Chania">
```

Try it Yourself »

Tip: A screen reader is a software program that reads the HTML code, and allows the user to "listen" to the content. Screen readers are useful for people who are visually impaired or learning disabled.

Image Size - Width and Height

You can use the `style` attribute to specify the width and height of an image.

Example

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

Try it Yourself »

Alternatively, you can use the `width` and `height` attributes:

Example

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

Try it Yourself »

The `width` and `height` attributes always define the width and height of the image in pixels.

Note: Always specify the width and height of an image. If width and height are not specified, the web page might flicker while the image loads.

Width and Height, or Style?

The `width`, `height`, and `style` attributes are all valid in HTML.

However, we suggest using the `style` attribute. It prevents styles sheets from changing the size of images:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  width: 100%;
}
</style>
</head>
<body>

<img src="html5.gif" alt="HTML5 Icon" width="128" height="128">
```

```
<img src="html5.gif" alt="HTML5 Icon" style="width:128px;height:128px;">

</body>
</html>
```

The **<a> tag (anchor tag)** in HTML is used to create a hyperlink on the webpage. This hyperlink is used to link the webpage to other web pages or some section of the same web page. It's either used to provide an absolute reference or a relative reference as its "href" value.

Syntax:

```
<a href = "link"> Link Name </a>
```

Attribute: The anchor tag contains many attributes which are listed below.

- **HTML <a> charset Attribute:** This attribute is used to specifies the character-set. It is not supported by HTML 5.
- **HTML <a> download Attribute:** It is used to specify the target link to download when the user clicks.
- **HTML <a> hreflang Attribute:** It is used to specify the language of the linked document.
- **HTML <a> media Attribute:** It is used to specify the linked media.
- **HTML <a> coords Attribute:** It is used to specify the coordinate of links. It is not supported by HTML 5.
- **HTML <a> name Attribute:** It is used to specify the anchor name. It is not supported by HTML 5 you can use the global **id attribute** instead.
- **HTML <a> rel Attribute:** It is used to specify the relation between the current document and the linked document.
- **HTML <a> shape Attribute:** It is used to specify the shape of the link. It is not supported by HTML 5.
- **HTML <a> type Attribute:** It is used to specify the type of links.
- **HTML <a> target Attribute:** It specifies the target link.
- **HTML <a> rev Attribute:** It is used to specify the relation between the linked document and the current document. It is not supported by HTML 5.

Example 1: In this example, the GeeksforGeeks HTML Tutorial page will open when you click on the GeeksforGeeks HTML Tutorial link.

- HTML

```
<!DOCTYPE html>

<html>
```

```
<body>

  <h2>Welcome to GeeksforGeeks HTML Tutorial</h2>

  <a href="https://www.geeksforgeeks.org/html-tutorials/">

     GeeksforGeeks HTML Tutorial

  </a>

</body>

</html>
```

The **<select> tag** in HTML is used to create a drop-down list. The <select> tag contains <option> tag to display the available option of drop-down list.
Note: The <select> tag is used in a form to receive user responses.
Syntax:

```
<select>
    <option>
    </option>
    ...
</select>
```

Attributes: The attributes of the <select> tag are listed below:

- **autofocus:** The HTML <select> autofocus Attribute is used to specify that the dropdown should automatically get focus when the page loads. It is a type of boolean attribute.
- **disabled:** The <select> disabled attribute is used to specify the select element is disabled. A disabled drop-down list is un-clickable and unusable. It is a boolean attribute.
- **form:** The HTML <select> form attribute is used to specify one or more forms that the <select> element belongs to.
- **multiple:** The HTML <select> multiple attribute is a Boolean Attribute. It specifies that the user is allowed to select more than one value that presents in <select> element.
- **name:** The HTML <select> name attribute is used to specify a name for the drop-down list. It is used to reference the form data after submitting the form or to reference the element in JavaScript.

- **required:** The HTML <select> required attribute is a Boolean attribute that is used to specify that the user should be selected value before submitting the Form.
- **size:** The HTML size attribute is used to specify the number of visible options in a drop-down list.

Example 1: In this example, we simply create a drop-down list in HTML.

- HTML

```
<!DOCTYPE html>

<html>



<body>

    <h2>Welcome To GeeksforGeeks</h2>

    <select>

        <option value="By the way">BTW</option>

        <option value="Talk to you later">TTYL</option>

        <option value="To be honest">TBH</option>

        <option value=" I don’t know">IDK</option>

    </select>

</body>



</html>
```

HTML tag is used as a generic container of inline elements. It is used for styling purpose to the grouped inline elements (using class and id attribute or inline style).

The tag does not have any default meaning or rendering.

The tag can be useful for the following task:

- To change the language of a part of the text.
- To change the color, font, background of a part of text using CSS
- To apply the scripts to the particular part of the text.

> Note: HTML is much similar as tag, but is used for block-level elements and tag is used for inline elements.

Syntax

1. ****Write your content here.....****

Following are some specifications about the HTML tag

Display	**Inline**
Start tag/End tag	Both Start and End tag
Usage	Styles and semantics

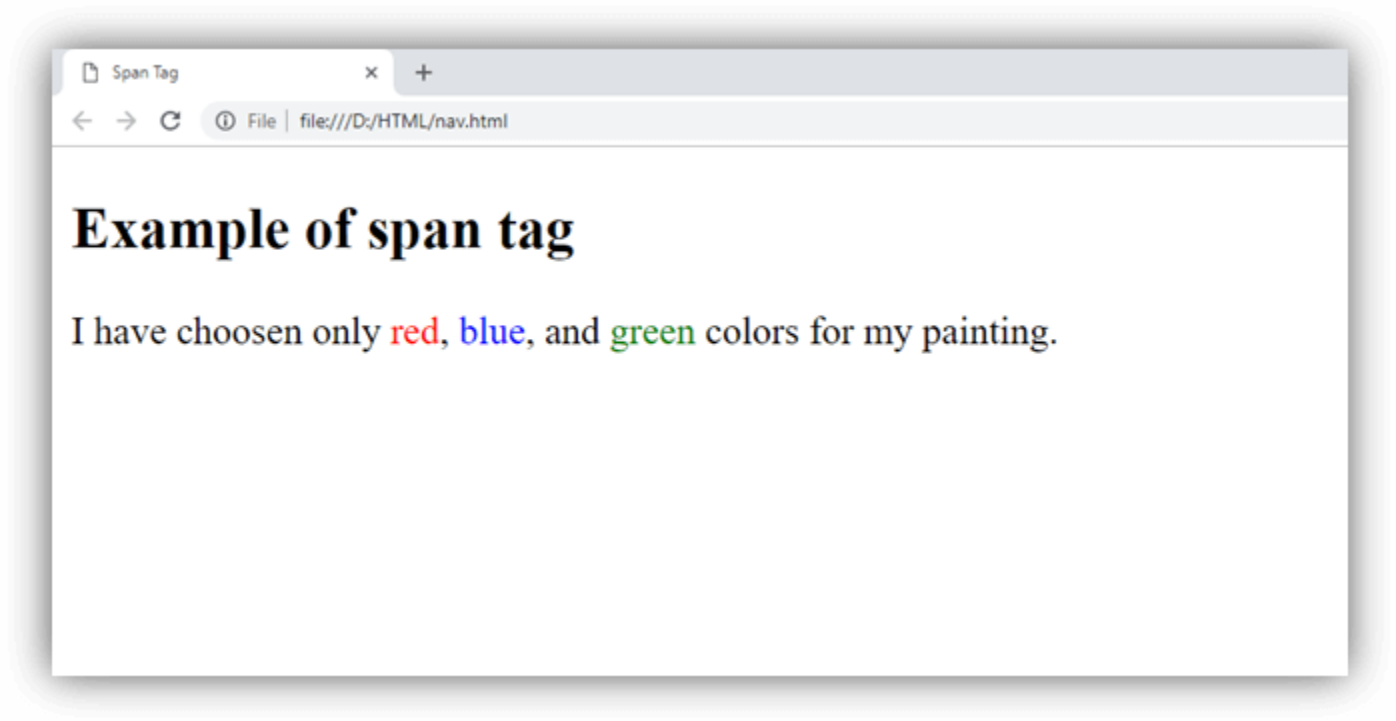
Example

```
<!DOCTYPE html>

<html>
<head>
 <title>Span Tag</title>
 </head>
<body>
  <h2>Example of span tag</h2>
  <p>I have choosen only
    <span style="color: red;">red</span>,
    <span style="color: blue;">blue</span>, and
    <span style="color: green;">green</span> colors for my painting.
```

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

Output:



Attribute:

Tag-specific attributes:

The tag does not contain any specific attribute in HTML.

Global attribute:

The tag supports the Global attributes in HTML.

Event attribute:

The tag supports the Event attributes in HTML.

Definition and Usage

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">`

Attributes

Global Attributes

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

Event Attributes

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

Ans 4

```
<table border=1>
    <caption>Table Caption</caption>
    <tr>
        <th>Heading1</th>
        <th>Heading2</th>
    </tr>
    <tr>
        <td>Row1 Col1 Data</td>
        <td>Row1 Col2 Data</td>
    </tr>
    <tr>
        <td>Row2 Col1 Data</td>
        <td>Row2 Col2 Data</td>
    </tr>
```

</table>

<table>	table tag
<caption>	optional table title
<tr>	table row
<th>	table header
<td>	table data element

Heading 1	Heading 2
Row1 Col1 Data	Row1 Col2 Data
Row2 Col1 Data	Row2 Col2 Data

```
<!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>Table</title>
</head>
<body>
    <table border="1" width= "50%" height= "50%">
        <tr>
            <th colspan="2">A</th>
            <th colspan="5">B</th>

        </tr>
        <tr>
                <th>C</th>
                <th>D</th>
                <th>E</th>
```

```
                <th>F</th>
                <th>G</th>
                <th>H</th>
                <th rowspan="3" align="center" valign="top">J</th>
        </tr>

        <tr>
                <td rowspan="2" align="center" valign="top">K</td>
                <th align="Center">M</th>
                <th colspan="2" align="Center">N</th>
                <th colspan="2">I</th>

        </tr>
        <tr>
            <th align="Center">L</th>
            <th align="Center">P</th>
            <th align="Center">O</th>
            <th align="Center">I</th>
            <th align="Center">H</th>

        </tr>
    </table>
</body>
</html>
```

Ans 5

The rowspan and colspan are <td> tag attributes. These are used to specify the number of rows or columns a cell should span. The rowspan attribute is for rows as well as the colspan attribute is for columns.

```
<!DOCTYPE html>
<html>
<head></head>
<body>
<table>
<tr>
<td colspan="add_numberof_cols"></td>
</tr>
<tr>
<td rowspan="add_numberof_rows ></td>
</tr>
</table>
</body>
</html>
```

These attributes have numeric values, for example, colspan=3 will span three columns.You can try to run the following code to work with the rowspan and colspan attribute in HTML

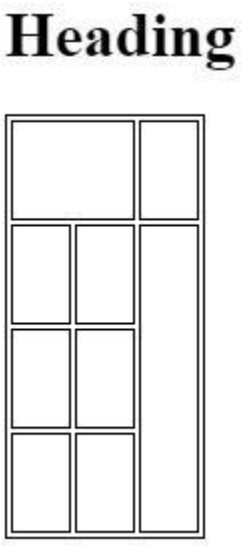
Example

```
<html>
   <head>
      <style>
         table, th, td {
            border: 1px solid black;
            width: 100px;
            height: 50px;
         }
      </style>
   </head>

   <body>
      <h1>Heading</h1>
      <table>
         <tr>
            <th colspan="2"></th>
            <th></th>
         </tr>
         <tr>
            <td></td>
            <td></td>
            <td rowspan="3"></td>
         </tr>
         <tr>
            <td></td>
            <td></td>
```

```
        </tr>
        <tr>
           <td></td>
           <td></td>
        </tr>
     </table>
  </body>
</html>
```

Output



Ans 6

- Metadata is data (information) about data.
- The <meta> tag provides metadata about the HTML document.
- <meta> tags always go inside the <head> element, and are typically used to specify character set, page description, keywords, author of the document, expiry dates and viewport settings.
- Metadata will not be displayed on the page.

- The metadata can be used by search engines (keywords), browsers (how to display content or reload page) or other web services.
- You can use <meta> tag to specify important keywords related to the document and later these keywords are used by the search engines while indexing your webpage for searching purpose.

Attribute	Value	Description
charset	character_set	Specifies the character encoding for the HTML document
name	author description keywords robots expires	Specifies a name for the metadata
http-equiv	content-type default-style refresh	Provides an HTTP header for the information/value of the content attribute
content	text	Gives the value associated with the http-equiv or name attribute
scheme	format/URI USA/Europe	Not supported in HTML5. Specifies a scheme to be

		used to interpret the value of the content attribute

Define keywords for search engines:

<meta name="keywords" content="HTML, CSS, JavaScript">

Define a description of your web page:

<meta name="description" content="Free Web tutorials for HTML and CSS">

Define the author of a page:

<meta name="author" content="John Doe">

Refresh document every 30 seconds:

<meta http-equiv="refresh" content="30">

Page redirection:

<meta http-equiv = "refresh" content = “10”; url = http://www.asoit.edu.in" />

Stop the browser from caching a page (disable browser cache):

<meta http-equiv="Cache-Control" content="no-store" />

Stop the page from being listed in search result :

<meta name="robots" content="noindex,nofollow">

Setting the viewport to make your website look good on all devices:

<meta name="viewport" content="width=device-width, initial-scale=1.0">

Sr.No.	Attribute & Description
1	**autoplay** This Boolean attribute if specified, the video will automatically begin to play back as soon as it can do so without stopping to finish loading the data.
2	**controls** If this attribute is present, it will allow the user to control video playback, including volume, seeking, and pause/resume playback.
3	**height** This attribute specifies the height of the video's display area, in CSS pixels.
4	**loop** This Boolean attribute if specified, will allow video automatically seek back to the start after reaching at the end.
5	**src** The URL of the video to embed. This is optional; you may instead use the <source> element within the video block to specify the video to embed.
6	**width** This attribute specifies the width of the video's display area, in CSS pixels.

- Most commonly used audio formats are **ogg, mp3** and **wav**.
- You can use <source> tag to specify media along with media type and many other attributes.

Attributes	**Attribute & Description**

autoplay	This Boolean attribute if specified, the audio will automatically begin to play back as soon as it can do so without stopping to finish loading the data.
Controls	It will allow the user to control audio playback, including volume, seeking, and pause/resume playback.
loop	This Boolean attribute if specified, will allow audio automatically seek back to the start after reaching at the end.
src	The URL of the audio to embed.

Ans 7

- **Frameset is used to** divide a browser screen into two or more HTML regions/sections.
- Tag used is the <FRAMESET>..</FRAMESET> tags.
- Each unique region is called a frame.
- Each frame can be loaded with a different document and hence**, allow multiple HTML documents to be seen concurrently.**
- FRAMESET tag is included after <HEAD> tag and We will not write <BODY> tag.
- 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.

Syntax:

<frameset cols = "pixels | % | *"> <frameset rows = "pixels | % | *">

frameset attributes

cols: The cols attribute is used to create vertical frames in a web browser. This attribute is basically used to define the no. of columns and their size inside the frameset tag. (Size may be in pixel or percentage)

rows: The rows attribute is used to create horizontal frames in the web browser. This attribute is used to define the no. of rows and their size inside the frameset tag. (Size may be in pixel **or** percentage **or** as a proportion of the remaining space by using '*'.)

border: This attribute of frameset tag defines the width of the border of each frame in pixels. Zero value is used for no border. (Size may be in pixel **or** percentage **or** as a proportion of the remaining space by using '*'.)

frameborder: This attribute of frameset tag is used to specify whether a three-dimensional border should be displayed between the frames or not for this use two values 0 and 1, where 0 defines no border and value 1 signifies for yes there will be a border.

framespacing: This attribute of frameset tag is used to specify the amount of spacing between the frames in a frameset. This can take any integer value as a parameter which basically denotes the value in pixel.

```
<frameset rows="10%, *, 10%" border="0">
    <frame  src="header.html">
```

```
    <frameset cols="20%,*,20%">
        <frame src="left.html">
        <frame src="maincontent.html"
name="main">
        <frame src="right.html">
    </frameset>
  <frame src="footer.html">
</frameset>
```

Ans 8

```
<!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>
    <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{10}" minlength="10"
maxlength="10"
                                   required placeholder="Enter Your Contact
No."></td>
                    </tr>
                    <tr> <th> <b>Landline no : (079-123456)</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>
```

Ans 9

What is search engine optimization?

Search engine optimization (SEO) is the art and science of getting pages to rank higher in search engines such as Google. Because search is one of the main ways in which people discover content online, ranking higher in search engines can lead to an increase in traffic to a website.

In Google and other search engines, the results page often features paid ads at the top of the page, followed by the regular results or what search marketers call the "organic search results". Traffic that comes via SEO is often referred to as "organic search traffic" to differentiate it from traffic that comes through paid search. Paid search is often referred to as search engine marketing (SEM) or pay-per-click (PPC).

The benefits of SEO

Search engine optimization is a key part of online marketing because search is one of the primary ways that users navigate the web.

Search results are presented in an ordered list, and the higher up on that list a site can get, the more traffic the site will tend to receive. For example, for a typical search query, the number one result will receive 40-60% of the total traffic for that query, with the number two and three results receiving significantly less traffic. Only 2-3% of searchers click beyond the first page of search results. Thus, even a small improvement in search engine rankings can result in a website receiving more traffic and potentially business.

Because of this, many businesses and website owners will try to manipulate the search results so that their site shows up higher on the search results page (SERP) than their competitors. This is where SEO comes in.

Chapter 3

Ans 1

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:

- Inline CSS
- Internal or Embedded CSS
- 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:

- html

```
<!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:

- html

```
<!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;
}
```

Below is the HTML file that is making use of the created external style sheet

- **link** tag is used to link the external style sheet with the html webpage.
- **href** attribute is used to specify the location of the external style sheet file.

- html

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="geeks.css"/>
    </head>
```

```
    <body>

        <div class = "main">

            <div class ="GFG">GeeksForGeeks</div>

            <div id ="geeks">

                A computer science portal for geeks

            </div>

        </div>

    </body>

</html>
```

Properties of CSS: Inline CSS has the highest priority, then comes Internal/Embedded followed by External CSS which has the least priority. Multiple style sheets can be defined on one page. If for an HTML tag, styles are defined in multiple style sheets then the below order will be followed.

- As Inline has the highest priority, any styles that are defined in the internal and external style sheets are overridden by Inline styles.
- Internal or Embedded stands second in the priority list and overrides the styles in the external style sheet.
- External style sheets have the least priority. If there are no styles defined either in inline or internal style sheet then external style sheet rules are applied for the HTML tags.

Supported Browser:

- Google Chrome
- Internet Explorer
- Firefox
- Opera
- Safari

CSS is the foundation of webpages, is used for webpage development by styling websites and web apps.You can learn CSS from the ground up by following this CSS Tutorial and CSS Examples.

Ans 2

Class	Id
We can apply a class to various elements so that it could be numerous times on a single page.	The Id is unique in a page, and we can only apply it to one specific element.
The class is assigned to an element and its name starts with "." followed by the name of the class.	The name of the Id starts with the "#" symbol followed by a unique id name.
We can attach multiple class selectors to an element.	We can attach only one ID selector to an element.

Syntax:
.class{
// declarations of CSS
}
	Syntax:
	#id{
	// declarations of CSS
	}

CSS Selectors

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

This page will explain the most basic CSS selectors.

The CSS element Selector

The element selector selects HTML elements based on the element name.

Example

Here, all <p> elements on the page will be center-aligned, with a red text color:

```
p {
  text-align: center;
  color: red;
}
```

ans 3

A Pseudo class in CSS is used to define the special state of an element. It can be combined with a CSS selector to add an effect to existing elements based on their states. For Example, changing the style of an element when the user hovers over it, or when a link is visited. All of these can be done using Pseudo Classes in CSS.

Note that pseudo-class names are not case-sensitive.
Syntax:

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

There are many Pseudo classes in CSS but the ones which are most commonly used are as follows:

- **:hover Pseudo-class:** This pseudo-class is used to add special effect to an element when our mouse pointer is over it. The below example demonstrates that when your mouse enters the box area, its background color changes from yellow to orange.
  Example:

- HTML

```
<!DOCTYPE html>

<html>

<head>

    <title>CSS transition-property property</title>

    <style>

    .box{

        background-color: yellow;

        width: 300px;

        height: 200px;

        margin: auto;

        font-size: 40px;

        text-align: center;

    }



    .box:hover{

        background-color: orange;

    }



    h1, h2{

        color: green;
```

```
        text-align: center;
    }
    </style>
</head>

<body>
    <h1>Geeks For Geeks</h1>
    <h2>:hover Pseudo-class</h2>
    <div class="box">
        My color changes if you hover over me!
    </div>
</body>
</html>
```

- **Output:**

- **:active Pseudo-class:** This pseudo-class is used to select an element which is activated when the user clicks on it. The following example demonstrates that when you click on the box, its background color changes for a moment.

- **:focus Pseudo-class:** This pseudo-class is used to select an element which is currently focused by the user. It works on user input elements used in forms and is triggered as soon as the user clicks on it. In the following example, the background color of the input field which is currently focused changes.
- **:visited Pseudo-class:** This pseudo-class is used to select the links which have been already visited by the user. In the following example, the color of the link changes once it is visited.

ans 4

The `position` property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

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;
}
```

Try it Yourself »

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).

Ans 5

The **Media query** in CSS is used to create a responsive web design. It means that the view of a web page differs from system to system based on screen or media types. The breakpoint specifies for what device-width size, the content is just starting to break or deform.
Media queries can be used to check many things:

- width and height of the viewport
- width and height of the device
- Orientation
- Resolution

A media query consist of a media type that can contain one or more expression which can be either true or false. The result of the query is true if the specified media matches the type of device the document is displayed on. If the media query is true then a style sheet is applied.

Syntax:

```
@media not | only mediatype and (expression) {
    // Code content
}
```

Example: This example illustrates the CSS media query with the different device-width for making it responsive.

- HTML

```
<!DOCTYPE html>
<html>

<head>
    <title>CSS media query</title>
    <style>
    body {
```

```
    text-align: center;
}

.gfg {
    font-size: 40px;
    font-weight: bold;
    color: green;
}

@media screen and (max-width:800px) {
    body {
        text-align: center;
        background-color: green;
    }
    .gfg {
        font-size: 30px;
        font-weight: bold;
        color: white;
    }
    .geeks {
```

```
            color: white;
        }
    }

    @media screen and (max-width:500px) {
        body {
            text-align: center;
            background-color: blue;
        }
    }
    </style>
</head>

<body>
    <div class="gfg">GeeksforGeeks</div>
    <div class="geeks">A computer science portal for geeks</div>
</body>

</html>
```

Output: From the output, we can see that if the max-width of the screen is reduced to 800px then the background color changes to green & if the max-width

of the screen is reduced to 500px then the background color will turn to blue. For the desktop size width, the background color will be white.

Media Types in CSS: There are many types of media types which are listed below:

- **all:** It is used for all media devices
- **print:** It is used for printer.
- **screen:** It is used for computer screens, smartphones, etc.
- **speech:** It is used for screen readers that read the screen aloud.

Features of Media query: There are many features of media query which are listed below:

- **color:** The number of bits per color component for the output device.
- **grid:** Checks whether the device is grid or bitmap.
- **height:** The viewport height.
- **aspect ratio:** The ratio between width and height of the viewport.
- **color-index:** The number of colors the device can display.
- **max-resolution:** The maximum resolution of the device using dpi and dpcm.
- **monochrome:** The number of bits per color on a monochrome device.
- **scan:** The scanning of output devices.
- **update:** How quickly can the output device modify.
- **width:** The viewport width.

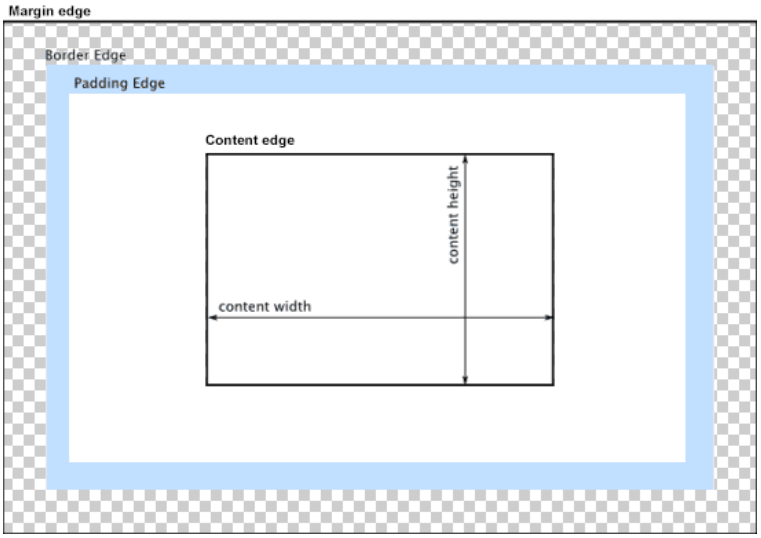
Supported Browsers: The browser supported by CSS *media query* are listed below:

- Chrome 21.0 and above
- Mozilla 3.5 and above
- Microsoft Edge 12.0
- Opera 9.0 and above
- Internet Explorer 9.0 and above
- Safari 4.0 and above

Ans 6

When laying out a document, the browser's rendering engine represents each element as a rectangular box according to the standard **CSS basic box model**. CSS determines the size, position, and properties (color, background, border size, etc.) of these boxes.

Every box is composed of four parts (or *areas*), defined by their respective edges: the *content edge*, *padding edge*, *border edge*, and *margin edge*.



Content area

The **content area**, bounded by the content edge, contains the "real" content of the element, such as text, an image, or a video player. Its dimensions are the *content width* (or *content-box width*) and the *content height* (or *content-box height*). It often has a background color or background image.

If the `box-sizing` property is set to `content-box` (default) and if the element is a block element, the content area's size can be explicitly defined with the `width`, `min-width`, `max-width`, `height`, `min-height`, and `max-height` properties.

Padding area

The **padding area**, bounded by the padding edge, extends the content area to include the element's padding. Its dimensions are the *padding-box width* and the *padding-box height*.

The thickness of the padding is determined by the `padding-top`, `padding-right`, `padding-bottom`, `padding-left`, and shorthand `padding` properties.

Border area

The **border area**, bounded by the border edge, extends the padding area to include the element's borders. Its dimensions are the *border-box width* and the *border-box height*.

The thickness of the borders are determined by the `border-width` and shorthand `border` properties. If the `box-sizing` property is set to `border-box`, the border area's size can be explicitly defined with the `width`, `min-width`, `max-width`, `height`, `min-height`, and `max-height` properties. When there is a background (`background-color` or `background-image`) set on a box, it extends to the outer edge of the border (i.e. extends underneath the border in z-ordering). This default behavior can be altered with the `background-clip` CSS property.

Margin area

The **margin area**, bounded by the margin edge, extends the border area to include an empty area used to separate the element from its neighbors. Its dimensions are the *margin-box width* and the *margin-box height*.

The size of the margin area is determined by the `margin-top`, `margin-right`, `margin-bottom`, `margin-left`, and shorthand `margin` properties. When margin collapsing occurs, the margin area is not clearly defined since margins are shared between boxes.

Finally, note that for non-replaced inline elements, the amount of space taken up (the contribution to the height of the line) is determined by the `line-height` property, even though the borders and padding are still displayed around the content.

Ans 7

CSS Colors

The color property in CSS is used to set the color of HTML elements. Typically, this property is used to set the background color or the font color of an element.

In CSS, we use color values for specifying the color. We can also use this property for the border-color and other decorative effects.

We can define the color of an element by using the following ways:

- RGB format.
- RGBA format.
- Hexadecimal notation.
- HSL.
- HSLA.
- Built-in color.

Let's understand the syntax and description of the above ways in detail.

RGB Format

RGB format is the short form of '**RED GREEN** and **BLUE**' that is used for defining the color of an HTML element simply by specifying the values of R, G, B that are in the range of 0 to 255.

The color values in this format are specified by using the **rgb()** property. This property allows three values that can either be in percentage or integer (range from 0 to 255).

This property is not supported in all browsers; that's why it is not recommended to use it.

Syntax

1. color: rgb(R, G, B);

RGBA Format

It is almost similar to RGB format except that **RGBA** contains **A (Alpha)** that specifies the element's transparency. The value of alpha is in the range **0.0 to 1.0**, in which **0.0** is for fully transparent, and **1.0** is for not transparent.

Syntax

1. color:rgba(R, G, B, A);

Hexadecimal notation

Hexadecimal can be defined as a six-digit color representation. This notation starts with the **# symbol** followed by six characters ranges from **0 to F**. In hexadecimal notation, the first two digits represent the **red (RR)** color value, the next two digits represent the **green (GG)** color value, and the last two digits represent the **blue (BB)** color value.

The black color notation in hexadecimal is #000000, and the white color notation in hexadecimal is #FFFFFF. Some of the codes in hexadecimal notation are #FF0000, #00FF00, #0000FF, #FFFF00, and many more.

Syntax

1. color:#(0-F)(0-F)(0-F)(0-F)(0-F)(0-F);

Short Hex codes

It is a short form of hexadecimal notation in which every digit is recreated to arrive at an equivalent hexadecimal value.

For example, #7B6 becomes #77BB66 in hexadecimal.

The black color notation in short hex is #000, and the white color notation in short hex is #FFF. Some of the codes in short hex are #F00, #0F0, #0FF, #FF0, and many more.

HSL

It is a short form of **Hue, Saturation,** and **Lightness**. Let's understand them individually.

Hue: It can be defined as the degree on the color wheel from 0 to 360. 0 represents red, 120 represents green, 240 represents blue.

Saturation: It takes value in percentage in which 100% represents fully saturated, i.e., no shades of gray, 50% represent 50% gray, but the color is still visible, and 0% represents fully unsaturated, i.e., completely gray, and the color is invisible.

Lightness: The lightness of the color can be defined as the light that we want to provide the color in which 0% represents black (there is no light), 50% represents neither dark nor light, and 100% represents white (full lightness).

Let's see the syntax of HSL in color property.

Syntax

```
1. color:hsl(H, S, L);
```

HSLA

It is entirely similar to HSL property, except that it contains **A (alpha)** that specifies the element's transparency. The value of alpha is in the range **0.0 to 1.0**, in which **0.0** indicates fully transparent, and **1.0** indicates not transparent.

Syntax

```
1. color:hsla(H, S, L, A);
```

Built-in Color

As its name implies, built-in color means the collection of previously defined colors that are used by using a name such as red, blue, green, etc.

Syntax

1. color: color-name;

Let's see the list of built-in colors along with their decimal and hexadecimal values.

S.no.	Color name	Hexadecimal Value	Decimal Value or rgb() value
1.	Red	#FF0000	rgb(255,0,0)
2.	Orange	#FFA500	rgb(255,165,0)
3.	Yellow	#FFFF00	rgb(255,255,0)
4.	Pink	#FFC0CB	rgb(255,192,203)
5.	Green	#008000	rgb(0,128,0)
6.	Violet	#EE82EE	rgb(238,130,238)
7.	Blue	#0000FF	rgb(0,0,255)

8.	Aqua	#00FFFF	rgb(0,255,255)
9.	Brown	#A52A2A	rgb(165,42,42)
10.	White	#FFFFFF	rgb(255,255,255)
11.	Gray	#808080	rgb(128,128,128)
12.	Black	#000000	rgb(0,0,0)

CSS background-color

The `background-color` property specifies the background color of an element.

Example

The background color of a page is set like this:

```
body {
  background-color: lightblue;
}
```

With CSS, a color is most often specified by:

- a valid color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

Look at CSS Color Values for a complete list of possible color values.

Other Elements

You can set the background color for any HTML elements:

Example

Here, the <h1>, <p>, and elements will have different background colors:

```
h1 {
  background-color: green;
}

div {
  background-color: lightblue;
}

p {
  background-color: yellow;
}
```

Opacity / Transparency

The `opacity` property specifies the opacity/transparency of an element. It can take a value from 0.0 - 1.0. The lower value, the more transparent:

opacity 1

opacity 0.6

opacity 0.3

opacity 0.1

Example

```
div {
  background-color: green;
  opacity: 0.3;
}
```

Transparency using RGBA

If you do not want to apply opacity to child elements, like in our example above, use **RGBA** color values. The following example sets the opacity for the background color and not the text:

100% opacity

60% opacity

30% opacity

10% opacity

You learned from our CSS Colors Chapter, that you can use RGB as a color value. In addition to RGB, you can use an RGB color value with an **alpha** channel (RGB**A**) - which specifies the opacity for a color.

An RGBA color value is specified with: rgba(red, green, blue, *alpha*).
The *alpha* parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

Tip: You will learn more about RGBA Colors in our CSS Colors Chapter.

Example

```
div {
  background: rgba(0, 128, 0, 0.3) /* Green background with 30% opacity */
}
```

The CSS Background Color Property

Property	Description
background-color	Sets the background color of an element

Chapter 4

Ans 1

What is JavaScript?

- A lightweight programming language ("scripting language")
- Used to create interactive user interface in a web page (e.g., menu, pop-up alert, windows, etc.)
- Manipulating web content dynamically
- Change the content and style of an element
- Replace images on a page without page reload
- Hide/Show contents
- Manipulating web content dynamically
- Dynamically **access and update** the **content**, **structure**, and **style** of a Web page.
- **Event Driven Programming** (ex: page load user click)
- Get information about a user's computer (ex: browser type)
- Perform calculations on user's computer (ex: form validation)
- Different brands or/and different versions of browsers may support different implementation of JavaScript.
- JScript is the Microsoft version of JavaScript.

Advantages of JavaScript

Speed : JavaScript is executed on the client side that's why it is very fast. Execute quickly because they do not require a trip to the server.

Simplicity : Simple and easy to run. Syntax, rules and regulations mostly same as C , Java Programming language.

Popularity: JavaScript is a very popular web language because it is used every where on the web.

Interoperability: JavaScript plays nicely with other languages and can be used in a wide range of applications.

Server Load : JavaScript reduce the server load as it executes on the client side. Although we can run java script also on server with NodeJS.

Browser Compatible: JavaScript supports all modern browsers. It can execute on any browser and produce same result.

Regular Updates: JavaScript updated annually by ECMA.

Rich interfaces: Gives the ability to create rich interfaces.

Ans 2

Client side scripting	**Server side scripting**

Source code is visible to user.	Source code is not visible to user because it’s output of server side is a HTML page.
It usually depends on browser and it’s version.	In this any server side technology can be use and it does not depend on client.
It runs on user’s computer.	It runs on web server.
There are many advantages link with this like faster response times, a more interactive application.	The primary advantage is it’s ability to highly customize, access rights based on user.
It does not provide security for data.	It provides more security for data.
It is a technique use in web development in which scripts runs on clients browser.	It is a technique that uses scripts on web server to produce a response that is customized for each clients request.
HTML, CSS and javascript are used.	PHP, Python, Java, Ruby are used.

End user can Turn off it from browser.

End user can no turn off it from browser.

Ans 3

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?";
}
```

Line Breaks

To display line breaks inside a popup box, use a back-slash followed by the character n.

Example

```
alert("Hello\nHow are you?");
```

ans 4

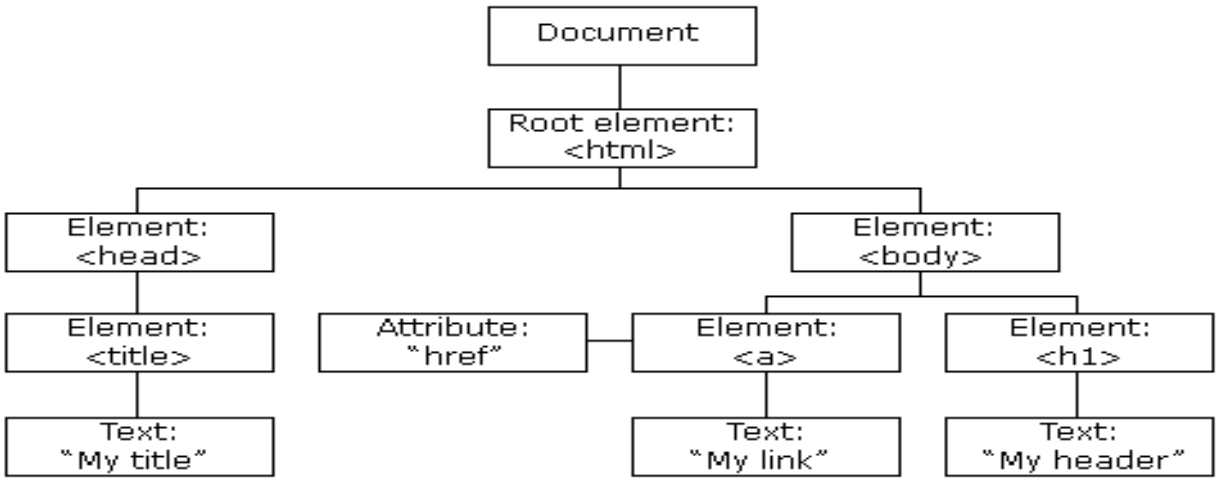
DOM – Introduction

- The Document Object Model is the standard way to accessing documents.
- DOM allows **scripts** to dynamically **access and update** the **content**, **structure**, and **style** of a document.
- Using JavaScript, you can create, modify and remove elements in the page dynamically.
- When a web page is loaded, the browser will consider all the tags/elements of page as an object and creates a Document Object Model of the page.
- The HTML DOM model is constructed as a tree of Objects.
- With the DOM, JavaScript can access and change all the elements of an HTML document.
- The nodes in a document make up the page's DOM tree, which describes the relationships among elements
- Nodes are related to each other through child-parent relationships
- A node may have multiple children, but only one parent

- The document node in a DOM tree is called the root node.

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 12.1: domtree.html -->
<!-- Demonstration of a document's DOM tree. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>DOM Tree Demonstration</title>
   </head>
   <body>
      <h1>An XHTML Page</h1>
      <p>This page contains some basic XHTML elements. We use the Firefox
         DOM Inspector and the IE Developer Toolbar to view the DOM tree
         of the document, which contains a DOM node for every element in
         the document.</p>
      <p>Here's a list:</p>
      <ul>
         <li>One</li>
         <li>Two</li>
         <li>Three</li>
      </ul>
   </body>
</html>
```

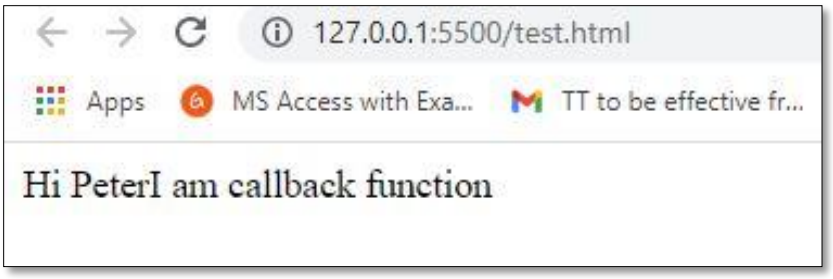


Ans 5

Callbacks in Javascript (Function as arguments)

- 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.
- A **callback** is a function which is called when a task is completed, thus helps in preventing any kind of blocking and a callback function allows other code to run in the meantime.
- Generally Callback function is used in Asynchronous communication
- function greet(name, callback) {
- document.write('Hi' + ' ' + name);
- callback();
- }
- function callMe() {
- document.write('I am callback function');
- }
- greet('Peter', callMe);
- 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.
- ❑ If you create a function to load an external resource (like a script or a file), you cannot use the content before it is fully loaded. This is the perfect time to use a callback.
- ❑ Ex: Function reading a file



Ans 6

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

```
<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>
```

MouseOver Event

```
<html>
<head>
<h1> Javascript Events </h1>
</head>
```

```
<body>
<script language="Javascript" type="text/Javascript">
    <!--
    function mouseoverevent()
    {
        alert("This is JavaTpoint");
    }
    //-->
</script>
<p onmouseover="mouseoverevent()"> Keep cursor over me</p>
</body>
</html>
```

Focus Event

```
<html>
<head> Javascript Events</head>
<body>
<h2> Enter something here</h2>
<input type="text" id="input1" onfocus="focusevent()"/>
<script>
<!--
    function focusevent()
    {
        document.getElementById("input1").style.background=" aqua";
    }
//-->
</script>
</body>
</html>
```

Keydown Event

```
<html>
<head> Javascript Events</head>
<body>
<h2> Enter something here</h2>
<input type="text" id="input1" onkeydown="keydownevent()"/>
```

```
<script>
<!--
    function keydownevent()
    {
        document.getElementById("input1");
        alert("Pressed a key");
    }
//-->
</script>
</body>
</html>
```

Load event

```
<html>
<head>Javascript Events</head>
</br>
<body onload="window.alert('Page successfully loaded');">
<script>
<!--
document.write("The page is loaded successfully");
//-->
</script>
</body>
</html>
```

Ans 7

Built-in Objects

- Built-in objects are not related to any Window or DOM object model.
- These objects are used for simple data processing in the JavaScript.

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.

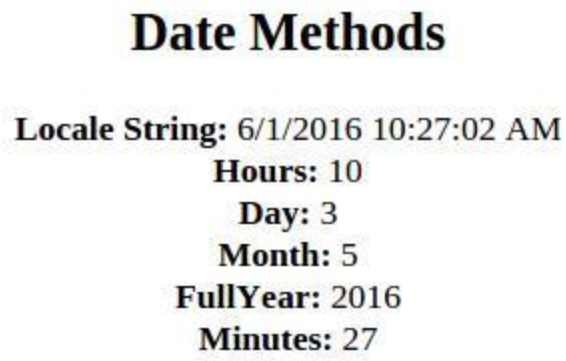
setHours()	Sets the hours.
setMinutes()	Sets the minutes.
setSeconds()	Sets the seconds.
setMilliseconds()	Sets the milliseconds.
setTime()	Sets the number of milliseconds since January 1, 1970 at 12:00 AM.
setMonth()	Sets the month.
toDateString()	Returns the date portion of the Date as a human-readable 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:



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:

Char At: a
CharCode At: 114
Index of: 7
Lower Case: careerride info
Upper Case: CAREERRIDE INFO

Ans 8

User Defined Objects

- JavaScript allows you to create your own objects.
- A javascript object is an entity having properties and method.
- Once you create object you can bind Properties and Methods with it and further as per your requirement you can retrieve that property values and invoke the methods.

  There are two ways to create object:

  (1) By Object Literal Notation

(2) By creating instance of Object (using New keyword)

Creating objects using **new Object()**

```
<script>
   var person = new Object();

   // Assign fields to object "person"

   person.firstName = “Sandip";
   person.lastName = “Patel";

   // Assign a method to object "person"

   person.sayHi = ()=>{
      alert("Welcome " + person.firstName + "  " + person.lastName)
   }

   person.sayHi();  // Call the method in "person"


 </script>
```

Ans 9

JavaScript Arrays

In this tutorial, you will learn about JavaScript arrays with the help of examples.

An array is an object that can store multiple values at once. For example,

```
const words = ['hello', 'world', 'welcome'];
```

Here, `words` is an array. The array is storing 3 values.

Create an Array

You can create an array using two ways:

1. Using an array literal

The easiest way to create an array is by using an array literal `[]`. For example,

```
const array1 = ["eat", "sleep"];
```

2. Using the new keyword

You can also create an array using JavaScript's `new` keyword.

```
const array2 = new Array("eat", "sleep");
```

In both of the above examples, we have created an array having two elements.

```
// empty array
const myList = [ ];

// array of numbers
const numberArray = [ 2, 4, 6, 8];
```

```
// array of strings
const stringArray = [ 'eat', 'work', 'sleep'];

// array with mixed data types
const newData = ['work', 'exercise', 1, true];
```

You can also store arrays, functions and other objects inside an array. For example,

```
const newData = [
    {'task1': 'exercise'},
    [1, 2 ,3],
    function hello() { console.log('hello')}
];
```

Access Elements of an Array

You can access elements of an array using indices **(0, 1, 2 ...)**. For example,

```
const myArray = ['h', 'e', 'l', 'l', 'o'];

// first element
console.log(myArray[0]);  // "h"

// second element
console.log(myArray[1]); // "e"
```

Array indexing in JavaScript

Add an Element to an Array

You can use the built-in method `push()` and `unshift()` to add elements to an array.

The `push()` method adds an element at the end of the array. For example,

```
let dailyActivities = ['eat', 'sleep'];

// add an element at the end
dailyActivities.push('exercise');

console.log(dailyActivities); //  ['eat', 'sleep', 'exercise']
```

The `unshift()` method adds an element at the beginning of the array. For example,

```
let dailyActivities = ['eat', 'sleep'];

//add an element at the start
dailyActivities.unshift('work');

console.log(dailyActivities); // ['work', 'eat', 'sleep']
```

Change the Elements of an Array

You can also add elements or change the elements by accessing the index value.

```
let dailyActivities = [ 'eat', 'sleep'];

// this will add the new element 'exercise' at the 2 index
dailyActivities[2] = 'exercise';

console.log(dailyActivities); // ['eat', 'sleep', 'exercise']
```

Suppose, an array has two elements. If you try to add an element at index 3 (fourth element), the third element will be undefined. For example,

```
let dailyActivities = [ 'eat', 'sleep'];

// this will add the new element 'exercise' at the 3 index
dailyActivities[3] = 'exercise';

console.log(dailyActivities); // ["eat", "sleep", undefined, "exercise"]
```

Basically, if you try to add elements to high indices, the indices in between will have undefined value.

Remove an Element from an Array

You can use the `pop()` method to remove the last element from an array. The `pop()` method also returns the returned value. For example,

```
let dailyActivities = ['work', 'eat', 'sleep', 'exercise'];
```

```
// remove the last element
dailyActivities.pop();
console.log(dailyActivities); // ['work', 'eat', 'sleep']

// remove the last element from ['work', 'eat', 'sleep']
const removedElement = dailyActivities.pop();

//get removed element
console.log(removedElement); // 'sleep'
console.log(dailyActivities);  // ['work', 'eat']
```

Run Code

If you need to remove the first element, you can use the `shift()` method. The `shift()` method removes the first element and also returns the removed element. For example,

```
let dailyActivities = ['work', 'eat', 'sleep'];

// remove the first element
dailyActivities.shift();

console.log(dailyActivities); // ['eat', 'sleep']
```

Array length

You can find the length of an element (the number of elements in an array) using the `length` property. For example,

```
const dailyActivities = [ 'eat', 'sleep'];

// this gives the total number of elements in an array
console.log(dailyActivities.length); // 2
```

Array Methods

In JavaScript, there are various array methods available that makes it easier to perform useful calculations.

Some of the commonly used JavaScript array methods are:

Method	Description
concat()	joins two or more arrays and returns a result
indexOf()	searches an element of an array and returns its position
find()	returns the first value of an array element that passes a test
findIndex()	returns the first index of an array element that passes a test
forEach()	calls a function for each element
includes()	checks if an array contains a specified element
push()	aads a new element to the end of an array and returns the new length of an array
unshift()	adds a new element to the beginning of an array and returns the new length of an array
pop()	removes the last element of an array and returns the removed element
shift()	removes the first element of an array and returns the removed element
sort()	sorts the elements alphabetically in strings and in ascending order
slice()	selects the part of an array and returns the new array
splice()	removes or replaces existing elements and/or adds new elements

Example: JavaScript Array Methods

```
let dailyActivities = ['sleep', 'work', 'exercise']
```

```
let newRoutine = ['eat'];

// sorting elements in the alphabetical order
dailyActivities.sort();
console.log(dailyActivities); // ['exercise', 'sleep', 'work']

//finding the index position of string
const position = dailyActivities.indexOf('work');
console.log(position); // 2

// slicing the array elements
const newDailyActivities = dailyActivities.slice(1);
console.log(newDailyActivities); // [ 'sleep', 'work']

// concatenating two arrays
const routine = dailyActivities.concat(newRoutine);
console.log(routine); // ["exercise", "sleep", "work", "eat"]
```

Working of JavaScript Arrays

In JavaScript, an array is an object. And, the indices of arrays are objects keys.

Since arrays are objects, the array elements are stored by reference. Hence, when an array value is copied, any change in the copied array will also reflect in the original array. For example,

```
let arr = ['h', 'e'];
let arr1 = arr;
arr1.push('l');

console.log(arr); // ["h", "e", "l"]
console.log(arr1); // ["h", "e", "l"]
```

You can also store values by passing a named key in an array. For example,

```
let arr = ['h', 'e'];
arr.name = 'John';

console.log(arr); // ["h", "e"]
console.log(arr.name); // "John"
console.log(arr['name']); // "John"
```

Run Code

Array indexing in JavaScript

However, it is not recommended to store values by passing arbitrary names in an array.

Hence in JavaScript, you should use an array if values are in ordered collection. Otherwise it's better to use object with `{ }`.

Ans 10

JSON

- ❑ JSON stands for JavaScript Object Notation.
- ❑ It has been extended from the JavaScript scripting language.
- ❑ The filename extension is **.json.**
- ❑ JSON Internet Media type is **application/json.**
- ❑ The JSON format is syntactically similar to the code for creating JavaScript objects.
- ❑ Because of this, a JavaScript program can easily convert JSON data into JavaScript objects.

- Since the format is text only, JSON data can easily be sent between computers, and used by any programming language.
- You can receive pure text from a server and use it as a JavaScript object.
- You can send a JavaScript object to a server in pure text format.
- You can work with data as JavaScript objects, with no complicated parsing and translations.
- JavaScript has a built in function for converting JSON strings into JavaScript objects:

JSON.parse()

JavaScript also has a built in function for converting an object into a JSON string:

JSON.stringify()

Uses of JSON

- It is used while writing JavaScript based applications that includes browser extensions and websites.
- JSON format is used for 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.
- It is a lightweight text-based interchange format.
- JSON is language independent.

JSON Syntax Rules

- JSON syntax is derived from JavaScript object notation syntax
- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

```
<script language = "javascript" >

   var object1 = { "language" : "Java", "author"  : "herbert schildt" };
   document.write("<h1>JSON with JavaScript example</h1>");
   document.write("<br>");
   document.write("<h3>Language = " + object1.language+"</h3>");
   document.write("<h3>Author = " + object1.author+"</h3>");

   var object2 = { "language" : "C++", "author"  : "E-Balagurusamy" };
   document.write("<br>");
   document.write("<h3>Language = " + object2.language+"</h3>");
   document.write("<h3>Author = " + object2.author+"</h3>");

   document.write("<hr />");
   document.write(object2.language + " programming language can be studied " +
"from book written by " + object2.author);
   document.write("<hr />");

</script>
```


Ans 11

Validation

Form Validation - Example

<script>

```
function validate() {
var form1 = document.getElementById("RegForm");
var name = form1.Name.value;
var email = form1.EMail.value;
var phone = form1.Telephone.value;
var what = form1.Subject.value;
var password = form1.Password.value;
var regEmail=/^\w+([\.-]?\w+)*@\w+([\.-
]?\w+)*(\.\w{2,3})+$/g;
var regPhone=/^\d{10}$/;
if (name == "") {
        window.alert("Please Enter Your Name");
        return false;
    }
    else if (email == "" || !regEmail.test(email)) {
        window.alert("Please enter a valid e-mail
address.");
        return false;
    }
    else if (password == "") {
        alert("Please enter your password");
        return false;
```

```
        }
        else if(password.length <6){
            alert("Password should be atleast 6 character
long");
            return false;
        }
        else if (phone == "" || !regPhone.test(phone)) {
            alert("Please enter valid phone number.");
            return false;
        }
        else if (what.selectedIndex == -1) {
            alert("Please enter your course.");
            return false;
        }
    }
</script>
<form name="RegForm" method="post"
id="RegForm"  onsubmit="return validate()" >
  <table>
    <tr>
        <td colspan="2"><h1 style="text-align:
center;">REGISTRATION FORM</h1></td>
```

```
      <td></td>
   </tr>
   <tr>
      <td>Name :</td>
      <td><input type="text" size="40" name="Name"
/> <br /></td>
   </tr>
   <tr>
      <td>E-mail Address:</td>
      <td><input type="text" size="40" name="EMail"
/><br></td>
   </tr>
   <tr>
      <td>Password:</td>
      <td><input type="text" size="40"
name="Password" /><br></td>
   </tr>
<tr>
      <td>Mobile:</td>
      <td><input type="text" size="40"
name="Telephone" /> <br></td>
   </tr>
   <tr>
```

```
        <td>Select Your Course : </td>
        <td><select type="text"name="Subject">
            <option>B.Tech</option>
            <option>M.Tech</option>
            <option>BCA</option>
            <option>MCA</option>
        </select> <br></td>
    </tr> <br/> <br/>
    <tr>
        <td>Comments:</td>
        <td><textarea cols="40" name="Comment">
</textarea> <br></td>
    </tr>
    <tr>
        <td></td>
        <td><input type="submit" value="Submit"
name="Submit" />  <input type="reset"
value="Reset" name="Reset" /> <br></td>
    </tr>
  </table>
</form>
```

Ans 12

```
<!DOCTYPE html>
```

```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>Practical 16</title>
</head>
<body>
    <script>
        var n1 = 0, n2 = 1, next_num, i;
        var num = parseInt(prompt("Enter Number Of Terms"));
        document.write("Fibonacci Series");
        for(i=1; i <= num; i++)
        {
            document.write("<br>"+n1);
            next_num = n1 + n2;
            n1 = n2;
            n2 = next_num;
        }
    </script>
</body>
</html>
```

Ans 13

```html
<!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>Practical 11</title>
</head>
<body id="mybody">
        <input type="button"  value="stop" id="btn1">
        <input type="button"  value="start" id="btn2">
</body>
<script>
       var mybody = document.getElementById("mybody");
       var btn1 = document.getElementById("btn1");
       var btn2 = document.getElementById("btn2");
       var color = ["Violet","Indigo","Blue","Green","Yellow","Orange","Red"];
       var i=0;
       var a = setInterval(()=>{
```

```
        mybody.style.backgroundColor=color[i];
        i++;
        if(i==color.length)
        {
            i=0;
        }
       },1000)
       btn1.addEventListener("click",()=>{
        clearInterval(a);
       });
       btn2.addEventListener("click",()=>{
        setInterval(()=>{
            mybody.style.backgroundColor=color[i];
        i++;
        if(i==color.length)
        {
            i=0;
        }
       },1000)
        });
</script>
</html>
```

Ans 14

```
<!DOCTYPE html>
<html>

<head>

    <title>

        Check a number is Prime or

        not using JavaScript

    </title>



    <script type="text/javascript">
```

```
        function p() {


            var n, i, flag = true;

            n = document.myform.n.value;

            n = parseInt(n)

            for(i = 2; i <= n - 1; i++)

                if (n % i == 0) {

                    flag = false;

                    break;

                }

            if (flag == true)

                alert(n + " is prime");

            else

                alert(n + " is not prime");

        }

    </script>
</head>

<body>

        <form name="myform">

            Enter the number:

            <input type="text" name=n value="">



            <br><br>
```

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

</html>
```

Ans 15

```
ww4ntent="width=device-width, initial-scale=1.0">
    <title>Practical 12</title>
</head>
<body>
    <p id="p1"> Silver Oak University</p>
</body>
<script>
    var p1 = document.getElementById("p1");
    p1.style.color="Blue";
    let i = 16;
    var a = setInterval(()=>{
        p1.style.fontSize=i+"px";
        i=i+4
        if(i>50)
        {
            clearInterval(a);
        }
    },1000);
</script>
</html>
```

Chapter 5

Ans 1

Node.js Advantages: Why use Node.js for developing web apps

- *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*

Where to Use Node.js?

- ❑ I/O bound Applications
- ❑ Data Streaming Applications
- ❑ Chat Application
- ❑ Real-time Applications
- ❑ JSON APIs based Applications

- ❑ Single Page Applications

Ans 2

Features of Node.js

Very Fast – Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.

Asynchronous – Non Blocking I/O, Asynchronous Programming

Single Threaded but Highly Scalable – Node.js uses a single threaded model **with event looping.** Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests.

Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.

No Buffering – Node.js cuts down the overall processing time while uploading audio and video files. Node.js applications never buffer any data.

Open source - Node.js has an open source community which has produced many excellent modules to add additional capabilities to Node.js applications.

Over 1,000,000 open source packages you can freely use..

Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)

1 Language for Frontend and Backend

Large Community Support

Ans 3

- NPM is the world's largest Software Library (Registry).
- NPM is also a software Package Manager and Installer.
- The registry contains over 800,000 code packages.
- Open-source developers use npm to share software.
- Many organizations also use npm to manage private development.
- NPM is a package manager for the JavaScript programming language.
- It is the default package manager for the JavaScript runtime environment Node.js.
- It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry.

- ❑ The registry is accessed via the client, and the available packages can be browsed and searched via the npm website.
- ❑ The package manager and the registry are managed by npm, Inc.
- ❑ NPM is included as a recommended feature in Node.js installer.
- ❑ npm consists of a command line client that interacts with a remote registry.
- ❑ It allows users to consume and distribute JavaScript modules that are available on the registry.

Managing Packages with NPM

- NPM is an acronym for Node.js Package Manager
- NPM provides a public package repository, a specification for building packages, and a command line tool for working with packages
  - http://www.npmjs.comnpm, Inc
- The company. develops and maintains NPM
- Node.js distributes the **npm** executable along with the **node** executable, but it's actually a separate program with its own versioning

Ans 4

- **REPL (READ, EVAL, PRINT, LOOP)** is a computer environment similar to Shell (Unix/Linux) and command prompt.
- It provides environment to run javascript from command prompt. (like shell script)
- Node comes with the REPL environment when it is installed
- It is useful in writing and debugging the codes.
- The work of REPL can be understood from its full form:

Read : It reads the inputs from users and parses it into JavaScript data structure. It is then stored to memory.
Eval : Takes and evaluates the data structure.
Print : The result is printed after the evaluation.
Loop : Loops the input command.

Getting Started with REPL:

- If you have node installed, then you also have the Node.js REPL.
- To start it, simply write **node** and press 'enter' to start the REPL.
- To come out of NODE REPL, press **ctrl+c** twice

```
C:\Users\Kazi Sir>node
Welcome to Node.js v16.15.1.
Type ".help" for more information.
```

- The **>** symbol lets you know that you can enter JavaScript code to be immediately evaluated.
- For an example, try adding two numbers in the REPL by typing 2 + 2 , you will get answer 4.

```
> 2+2
4
```

Simple Expression

Let's try a simple mathematics at the Node.js REPL command prompt -

```
$ node
> 1 + 3
4
> 1 + ( 2 * 3 ) - 4
3
>
```

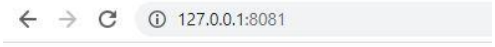
Ans 5

Serving HTML Files With Node.js

```
let http = require('http');
let fs = require('fs');
http.createServer((request, response)=>{
    response.writeHead(200, {
        'Content-Type': 'text/html'
    });
    fs.readFile('./sample.html',(error, data)=> {
        if (error) {
            response.writeHead(404);
            respone.write('File not found!');
        } else {
            response.write(data);
        }
        response.end();
    });
}).listen(8081);
```

Sample.html **Output : Run from local host**

```
<body>
<h2>An Unordered HTML List</h2>
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
<h2>An Ordered HTML List</h2>
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
</body>
```



An Unordered HTML List

- Coffee
- Tea
- Milk

An Ordered HTML List

1. Coffee
2. Tea
3. Milk

Ans 6

Rendering JSON Data

1) **JSON stands for JavaScript Object Notation. It is one of the most widely used formats for exchanging information across applications**
2) **var fs = require('fs');**

```
3)    fs.readFile('./world.json','utf-
   8',(error,data)=>{
4)
5)        var jsondata = JSON.parse(data);
6)        //console.log(jsondata);
7)        console.log(jsondata[0].cm.Maharashtr
   a);
8)        console.log(jsondata[1].famous_cities[
   2]);
9)    });
```

3) Output

```
ondemo> node test2.js
Shinde
Washington
```

- **4) JavaScript has a built in function for converting JSON strings into JavaScript objects:**

JSON.parse()

JavaScript also has a built in function for converting an object into a JSON string:

JSON.stringify()

[{

"country_id" : "1",

"cname": "INDIA",

"capital": "New Delhi",

"famous_cities":
["Ahmedabad","Hyderabad","Chennai","Mumbai"
,"Banglore","Kolkata"],

"pm": "Narendra Modi",

"cm": {"Gujarat":"Bhupendra Patel", "UP":"Yogi
Aditya Nath", "MP":"Shivraj
Singh","Maharashtra":"Shinde"}

},

{

"country_id" : "2",

"cname": "USA",

"capital": "Washington",

"famous_cities":
["Chicago","Callifornia","Washington","Newyork"
],

"pm": "Joe Biden",

"cm": {"Alabama":"Michel bishop",
"texas":"Peter john",
"michigan":"Ronaldo","Washington":"Mr.Jeorge
"}

},

{

"country_id" : "3",

"cname": "China",

"capital": "Beijing",

"famous_cities": ["Shanghai","Beijing","Chengdu","Xian"],

"pm": "Jin Ping",

"cm": {"Hainan":" Hainan1", "Hebel":" Hebel1", "Hubel":" Hubel1"}

}]

Ans 7

Create Server in Node JS

Create **.js** file – write below code and Run it on Visual Studio Terminal

```
var http = require('http');

http.createServer((req,res)=>{

    res.end("Welcome in ASOIT");

}).listen(8081);
```

```
> node test6.js
```

Output :

Ans 8

Asynchronous communication in Node JS

Code for reading a file asynchronously (non-blocking code) in Node.js.

Create a text file **inputfile1.txt** with some content.

```
// Write a JavaScript code
var fs = require("fs");

fs.readFile('inputfile1.txt', function (ferr, filedata) {
    if (ferr) return console.error(ferr);
    console.log(filedata.toString());
});
console.log("End of Program execution");
```

- *fs* library is loaded to handle file-system related operations.
- The readFile() function is asynchronous and control return immediately to the next instruction in the program while the function keep running in the background.
- A callback function is passed which gets called when the task running in the background are finished.

Ans 9

Routing in Node JS

What is Routing?

- Routing defines the way in which the client requests are handled by the application endpoints.
- Purpose of routing is to generate appropriate response for incoming client request.
- We can implement routing in node.js also with framework. (Express framework)

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.
- Create **.js** file – write below code and Run it on Visual Studio Terminal
- Routing Example:
- var http = require('http');
- var fs = require('fs');
- http.createServer((req,res)=>{
- res.writeHead(200,{'Content-Type':'text/html'});
- if(req.url == "/")
- {
- res.end("<h1> Welcome in Home page");
- }
- else if(req.url == "/pr2")

```
    {
    fs.readFile('./pr2.html',(err,data)=>{
        if(err) throw error;
        res.end(data);
    })
    }
    else if(req.url == "/pr9")
    {
    fs.readFile('./pr9.html',(err,data)=>{
        if(err) throw error;
        res.end(data);
    })
    }
    else
    {
    res.end("Page not found");
    }
}).listen(8081);
```