

# Web-Programming-Module-1-Important-Topics-PYQs

🔗 For more notes visit

<https://rtpnotes.vercel.app>

- Web-Programming-Module-1-Important-Topics-PYQs
- Important Topics
  - 1. Define URI and URLs
    - URI (Uniform Resource Identifier)
    - URL (Uniform Resource Locator)
    - Key Differences Between URI and URL:
  - 2. Internal Linking and External Linking
    - 1. Internal Linking
      - Example of Internal Linking:
    - 2. External Linking
      - Example of External Linking:
  - 3. How to create tables in HTML
    - Basic Structure of an HTML Table:
      - Example of a Simple Table:
        - Explanation of Tags:
  - 4. Different HTML5 form input elements
    - 1. Element
      - Example of Common Input Types:
    - 2. Element</x-turndown>
    - 3. and Elements
    - 4. Element
  - 5. Page structure elements
    - 1. html Tag
    - 2. head Tag

- 3. title Tag
- 4. body Tag
- 5. h1 to h6 Tags
- 6. p Tag
- 7. a (Anchor) Tag
- 8. img Tag
- 9. ul, ol, and li Tags
- 10. div Tag
- 11. span Tag
- 12. form Tag
- 13. br Tag
- 14. strong and em Tags
- 15. button Tag
- 6. Rowspan and Colspan
  - Understanding rowspan and colspan in HTML Tables
  - Step 1: Basics of Tables Without rowspan or colspan
  - Step 2: Using colspan (Merging Cells Across Columns)
  - Step 3: Using rowspan (Merging Cells Across Rows)
  - Step 4: Combining rowspan and colspan
  - Step 5: Visualize a Complex Table Layout
- Previous Year Questions
  - 1. Why do you call MIME as an extension feature? Justify with suitable statements.
  - 2. What is the purpose of the attributes autoplay, src, controls when used with audio tag?
    - Purpose of Attributes in the < audio> Tag:
    - 1. autoplay:
    - 2. src:
    - 3. controls:
  - 3. What is a URL? Explain the different parts of URL with example.
    - What is a URL?
    - Parts of a URL:
      - 1. Protocol (Scheme):
      - 2. Domain Name:

- 3. Port (Optional):
- 4. Path:
- 5. Query String (Optional):
- 6. Fragment (Optional):
- Example of a Full URL:
- 4. Describe World Wide Web and its significance.
  - Key Components of the World Wide Web:
  - Significance of the World Wide Web:
- 5. Differentiate between block tags and inline tags. Illustrate with suitable examples.
  - Block Tags
    - Example:
  - Inline Tags
    - Example:
  - Key Differences:
  - Visual Analogy:
- 6. Write the equivalent HTML code to implement the following in a web page:
  - Explanation for the Exam:
- 7. Explain following html tags with proper example.
  - 1. `</x-turndown>`
  - 3.
- 8. Design a webpage that displays the following table.
  - Explanation of the Table:
  - Output
- 9. Write down the general format of an HTTP request and an HTTP response. What is the purpose of the following HTTP headers? Also, identify whether they are included with an HTTP header/response or both. i. host ii. last-modified
  - General Format of an HTTP Request
    - Example of an HTTP Request
  - General Format of an HTTP Response
    - Example of an HTTP Response
    - i. Host
    - ii. Last-Modified

- 10. How will you navigate between sections in the same web page? Illustrate with appropriate example?
  - Example
  - How It Works:
  - Output in the Browser:
- 11. Create a HTML form for registering to a jobsite which includes fields to
  - Output
- 12. Develop HTML code to create the following web page?
  - Output
- 13. Give the syntax to create hyperlinks in HTML? List any 3 target attributes and their purpose associated with hyperlinks? Also how will you identify an active link, visited link and unvisited link?
  - Syntax to Create Hyperlinks in HTML
    - Example:
  - Target Attributes and Their Purpose
  - Identifying Active, Visited, and Unvisited Links
    - Example of Styling Links:
- 14. Write the equivalent HTML code to implement the following in a web page:
  - i. Flowers.jpg image
  - ii. Hyperlink to "birds.jpg"
  - iii. Unordered List with Tea, Coffee, and Milk
  - Complete HTML Code:
- 15. Write the HTML code for obtaining the following table.
- 16. Explain list tags with example.
  - 1. Ordered List (< ol>)
    - Example of an Ordered List:
  - 2. Unordered List (< ul>)
    - Example of an Unordered List:
- 17. Write the HTML code for obtaining the following registration form, where the course field contains the options BCA, BBA, B. Tech, MBA, MCA, M. Tech
  - Output
- 18. Explain HTTP and its significance. Describe the request and response phases in HTTP.

- Significance of HTTP:
  - HTTP Request and Response Phases
    - 1. HTTP Request Phase
    - 2. HTTP Response Phase
- 19. Explain MIME and its types with examples. Describe why should MIME type information be essentially included in HTTP responses.
  - MIME (Multipurpose Internet Mail Extensions) and Its Significance
  - Types of MIME Types
    - 1. Text
    - 2. Image
    - 3. Audio/Video
    - 4. Application
    - 5. Multipart
  - Why MIME Type Information Should Be Included in HTTP Responses
    - 1. Correct Content Rendering
    - 2. Content-Type Negotiation
    - 3. Proper Handling of Different Data Formats
    - 4. Efficient File Transfers
  - Example of HTTP Response with MIME Type:
- 20. Write HTML code to design a web page to create a table with 5 rows and 3 columns for entering the mark list of 4 students. Assume suitable headings for each column.

# Important Topics

## 1. Define URI and URLs

### URI (Uniform Resource Identifier)

A **URI** is a string of characters used to uniquely identify a resource on the internet or in a local system. It provides a way to identify resources, which may or may not be accessible over the internet.

- **General Definition:** A URI can refer to any resource, including a webpage, file, image, or even abstract concepts (like a specific section of a document or a software function).

- **Components:** A URI might consist of a **scheme** (like `http`, `ftp`, etc.), **authority** (like domain name or IP address), and an optional **path** or **query**.

### Example of URI:

- `http://example.com/resource`: This is a URI identifying a resource located at `http://example.com/resource`.

## URL (Uniform Resource Locator)

A **URL** is a specific type of URI that not only identifies a resource but also provides the means to **locate** it, often specifying a protocol (such as HTTP or FTP) to access the resource over a network.

- **Definition:** A URL is a URI that provides information about how to access the resource, typically including the protocol (HTTP/HTTPS), domain, and optional path/query parameters.

### Example of URL:

- `https://www.example.com/index.html`: This is a URL where `https` is the protocol, `www.example.com` is the domain, and `/index.html` is the path to the resource.

## Key Differences Between URI and URL:

1. **URI:** A URI is a broader term that refers to any identifier, whether it provides a way to locate the resource or not.
2. **URL:** A URL is a type of URI that not only identifies but also specifies how to find and access the resource using a protocol (like HTTP or FTP).



## 2. Internal Linking and External Linking

### 1. Internal Linking

**Internal linking** refers to linking between different pages or sections within the same website or domain. It helps users navigate through the different parts of the same website.

### Example of Internal Linking:

```
<a href="about.html">About Us</a> <!-- Linking to another page within the same website -->
```

- In this example, the link points to a page named `about.html` which is within the same website.



## 2. External Linking

**External linking** refers to linking to a different website or domain from your webpage. These links connect your site to content located on other websites.

**Example of External Linking:**

```
<a href="https://www.example.com">Visit Example Website</a> <!-- Linking to an external website -->
```

- In this example, the link points to an external website `https://www.example.com`.



## 3. How to create tables in HTML

In HTML, a **table** is used to display data in a structured way with rows and columns. The table element is represented using the `<table>` tag, and other tags are used to define the table's content, such as headers, rows, and cells.

**Basic Structure of an HTML Table:**

1. `<table>` : Defines the table itself.
2. `<tr>` : Defines a table row.
3. `<th>` : Defines a table header cell (for headings).
4. `<td>` : Defines a table data cell (for data).



**Example of a Simple Table:**

```

<table border="1">
  <tr>
    <th>Heading 1</th>
    <th>Heading 2</th>
    <th>Heading 3</th>
  </tr>
  <tr>
    <td>Row 1, Cell 1</td>
    <td>Row 1, Cell 2</td>
    <td>Row 1, Cell 3</td>
  </tr>
  <tr>
    <td>Row 2, Cell 1</td>
    <td>Row 2, Cell 2</td>
    <td>Row 2, Cell 3</td>
  </tr>
</table>

```

### Explanation of Tags:

- `<table>` : Defines the entire table.
- `<tr>` : Defines a table row.
- `<th>` : Defines a table header cell (bold and centered by default).
- `<td>` : Defines a table data cell.



## 4. Different HTML5 form input elements

### 1. `<input>` Element

The `<input>` element is used to create interactive controls in a form, such as text fields, checkboxes, radio buttons, and buttons. The type attribute defines the input type.

#### Example of Common Input Types:

```

<form>
  <!-- Text Input -->
  <label for="name">Name:</label>
  <input type="text" id="name" name="name">

```



```

<!-- Password Input -->
<label for="password">Password:</label>
<input type="password" id="password" name="password">

<!-- Checkbox Input -->
<label for="subscribe">Subscribe:</label>
<input type="checkbox" id="subscribe" name="subscribe">

<!-- Radio Buttons -->
<label for="male">Male:</label>
<input type="radio" id="male" name="gender" value="male">
<label for="female">Female:</label>
<input type="radio" id="female" name="gender" value="female">

<!-- Submit Button -->
<input type="submit" value="Submit">
</form>

```



## 2. <textarea> Element

The <textarea> element is used for multi-line text input, such as comments or long descriptions.

```

<form>
  <label for="message">Message:</label>
  <textarea id="message" name="message" rows="4" cols="50"></textarea>
</form>

```



## 3. <select> and <option> Elements

The <select> element is used to create a dropdown list. The <option> element defines the options within the dropdown.

```

<form>
  <label for="country">Country:</label>

```

```
<select id="country" name="country">
  <option value="usa">USA</option>
  <option value="canada">Canada</option>
  <option value="uk">UK</option>
</select>
</form>
```



## 4. <button> Element

The `<button>` element is used to create clickable buttons. It can be used in forms for submitting data or triggering JavaScript functions.

```
<form>
  <button type="submit">Submit Form</button>
  <button type="reset">Reset Form</button>
</form>
```



## 5. Page structure elements

HTML (Hypertext Markup Language) uses **tags** to structure content on a webpage. Each tag tells the browser how to display the content.

### 1. html Tag

- What It Does: Defines the beginning and end of an HTML document.
- Use: Wraps all the content on the page.

#### Syntax

```
<html>
  <!-- All content of the webpage goes here -->
</html>
```

### 2. head Tag

- **What It Does:** Contains meta-information about the webpage, like its title and links to stylesheets or scripts.
- **Use:** It doesn't display content directly on the page, but it helps set up the page's structure.

**Example:**

```
<head>
  <title>My Webpage</title>
  <link rel="stylesheet" href="styles.css">
</head>
```

### 3. title Tag

- **What It Does:** Defines the title of the webpage that appears in the browser tab.

**Example**

```
<title>My First Webpage</title>
```

### 4. body Tag

- **What It Does:** Contains the visible content of the webpage. Everything inside the `<body>` tag appears on the screen.

**Example:**

```
<body>
  <h1>Welcome to My Webpage</h1>
  <p>This is a paragraph of text.</p>
</body>
```

### 5. h1 to h6 Tags

- **What It Does:** Headings that define the title or section headers. `<h1>` is the largest (most important), and `<h6>` is the smallest.

**Example:**

```
<h1>Main Heading</h1>  
<h2>Subheading</h2>  
<h3>Smaller Subheading</h3>
```

## 6. p Tag

- **What It Does:** Defines a paragraph of text.

**Example:**

```
<p>This is a simple paragraph of text.</p>
```

## 7. a (Anchor) Tag

- **What It Does:** Creates a hyperlink to another webpage or location.

**Example:**

```
<a href="https://www.example.com">Click here to visit Example.com</a>
```

**href :** The URL to which the link points.

## 8. img Tag

- **What It Does:** Embeds an image in the webpage.

**Example**

```

```

- **src :** The source (file path) of the image.
- **alt :** Alternative text for the image (important for accessibility).
- **width and height :** Dimensions of the image.

## 9. ul, ol, and li Tags

- **What It Does:** Creates lists. `<ul>` is for unordered (bullet) lists, and `<ol>` is for ordered (numbered) lists. `<li>` represents each list item.

## Unordered List (bullets):

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
```

## Ordered List (numbers):

```
<ol>
  <li>First Item</li>
  <li>Second Item</li>
  <li>Third Item</li>
</ol>
```

## 10. div Tag

- **What It Does:** A container used to group HTML elements together, often for styling purposes with CSS.

```
<div>
  <h2>Section Title</h2>
  <p>This is some content inside a div.</p>
</div>
```

## 11. span Tag

- **What It Does:** A smaller container for inline elements (like text), often used to style or manipulate a specific part of the text.

### Example

```
<p>This is <span style="color:red;">important</span> text.</p>
```

This is **important** text.

## 12. form Tag

- **What It Does:** Creates a form to collect user input, such as text fields, checkboxes, and buttons.

#### Example

```
<form action="/submit" method="POST">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name">
  <input type="submit" value="Submit">
</form>
```

Name:

### 13. br Tag

- **What It Does:** Inserts a line break (used to move text to the next line).

#### Example

```
<p>This is the first line.<br>This is the second line.</p>
```

This is the first line.

This is the second line.

### 14. strong and em Tags

- **What It Does:** These tags are used for emphasizing text. `<strong>` makes text bold, and `<em>` makes text italic.

#### Example

```
<p>This is <strong>bold text</strong> and this is <em>italic text</em>.</p>
```

This is **bold text** and this is *italic text*.

### 15. button Tag

- **What It Does:** Creates a clickable button.

## Example

```
<button type="button">Click Me</button>
```

Click Me



## 6. Rowspan and Colspan

### Understanding rowspan and colspan in HTML Tables

In HTML, tables are used to display data in rows and columns. The attributes `rowspan` and `colspan` allow you to merge cells across rows or columns to organize data more effectively.

#### Step 1: Basics of Tables Without rowspan or colspan

Here's a basic table with no merging of cells:

```
<table border="1">
  <tr>
    <th>Header 1</th>
    <th>Header 2</th>
  </tr>
  <tr>
    <td>Row 1, Column 1</td>
    <td>Row 1, Column 2</td>
  </tr>
  <tr>
    <td>Row 2, Column 1</td>
    <td>Row 2, Column 2</td>
  </tr>
</table>
```

**Output:**

| Header 1        | Header 2        |
|-----------------|-----------------|
| Row 1, Column 1 | Row 1, Column 2 |
| Row 2, Column 1 | Row 2, Column 2 |

Each cell ( `<td>` or `<th>` ) occupies a single row and a single column.



## Step 2: Using `colspan` (Merging Cells Across Columns)

The `colspan` attribute merges cells horizontally (across columns).

Example:

```
<table border="1">
  <tr>
    <th colspan="2">Header spanning 2 columns</th>
  </tr>
  <tr>
    <td>Row 1, Column 1</td>
    <td>Row 1, Column 2</td>
  </tr>
</table>
```

Output:

| Header spanning 2 columns |                 |
|---------------------------|-----------------|
| Row 1, Column 1           | Row 1, Column 2 |



## Step 3: Using `rowspan` (Merging Cells Across Rows)

The `rowspan` attribute merges cells vertically (across rows).

Example:

```
<table border="1">
  <tr>
    <th>Header 1</th>
    <th>Header 2</th>
  </tr>
  <tr>
    <td rowspan="2">Row 1 & 2, Column 1</td>
```



```

    <td>Row 1, Column 2</td>
  </tr>
  <tr>
    <td>Row 2, Column 2</td>
  </tr>
</table>

```

### Output:

| Header 1            | Header 2        |
|---------------------|-----------------|
| Row 1 & 2, Column 1 | Row 1, Column 2 |
|                     | Row 2, Column 2 |

- The first `<td>` spans across **2 rows**, merging two cells into one.



## Step 4: Combining `rowspan` and `colspan`

You can use `rowspan` and `colspan` together to create complex table layouts.

Example:

```

<table border="1">
  <tr>
    <th colspan="2">Header spanning 2 columns</th>
    <th rowspan="2">Header spanning 2 rows</th>
  </tr>
  <tr>
    <td>Row 1, Column 1</td>
    <td>Row 1, Column 2</td>
  </tr>
  <tr>
    <td colspan="2">Row 2, Columns 1 & 2</td>
    <td>Row 2, Column 3</td>
  </tr>
</table>

```

### Output:

| Header spanning 2 columns |                 | Header spanning 2 rows |
|---------------------------|-----------------|------------------------|
| Row 1, Column 1           | Row 1, Column 2 |                        |
| Row 2, Columns 1 & 2      |                 | Row 2, Column 3        |

Here:

- The first `<th>` spans across **2 columns**.
- The second `<th>` spans across **2 rows**.
- In the third row, a `<td>` spans **2 columns**.



## Step 5: Visualize a Complex Table Layout

Now, let's create a more detailed table combining multiple `rowspan` and `colspan` for better data representation.

```
<table border="1">
  <tr>
    <th rowspan="2">Name</th>
    <th colspan="2">Marks</th>
    <th rowspan="2">Grade</th>
  </tr>
  <tr>
    <th>Math</th>
    <th>Science</th>
  </tr>
  <tr>
    <td>John</td>
    <td>85</td>
    <td>90</td>
    <td>A</td>
  </tr>
  <tr>
    <td>Jane</td>
    <td>88</td>
    <td>92</td>
    <td>A</td>
  </tr>
</table>
```

```
</tr>  
</table>
```

Output:

| Name | Marks |         | Grade |
|------|-------|---------|-------|
|      | Math  | Science |       |
| John | 85    | 90      | A     |
| Jane | 88    | 92      | A     |



## Previous Year Questions

### 1. Why do you call MIME as an extension feature? Justify with suitable statements.

- MIME stands for **Multipurpose Internet Mail Extensions**, and it's called an **extension feature** because it adds extra capabilities to the original email system, allowing it to handle more than just plain text.
- In simple terms, MIME helps emails and web pages support things like images, videos, and other files, which the old email system couldn't handle.

#### Features of MIME

##### 1. Support for More Than Just Text:

1. Originally, emails could only send plain text (just words). MIME allows emails and web pages to send all kinds of files like images, audio, and videos. This makes emails much more useful.
2. **Example:** With MIME, you can send an image or a PDF file in an email, which was impossible with the old system.

##### 2. Content-Type: MIME adds a special label called **Content-Type**. This label tells the system what kind of file is being sent, so the email program knows if it's text, an image, or something else.

1. **Example:** If you're sending an image, MIME will use `Content-Type: image/jpeg` to tell the system it's a JPEG image.

### 3. Encoding Files:

1. MIME also helps to turn files into a simple text format that can be safely sent over email.
2. **Example:** When you send a picture in an email, MIME turns it into a text format so it doesn't get lost during the transmission.

### 4. Multipart Messages:

1. MIME allows an email to contain multiple parts. For example, you can send an email with both a text message and an image in the same email.
2. **Example:** An email could have a text body, a picture attached, and maybe even a video—all in one email, thanks to MIME.



## 2. What is the purpose of the attributes autoplay, src, controls when used with audio tag?

### Purpose of Attributes in the < audio> Tag:

The `<audio>` tag in HTML is used to embed audio content (such as music, sound effects, etc.) into a webpage. It comes with several useful attributes to control how the audio behaves.

#### 1. autoplay:

- **Purpose:** This attribute tells the browser to start playing the audio as soon as it is ready, without needing the user to click the play button.
- **Example:** If you want an audio file to play automatically when the page loads, you can use the `autoplay` attribute.

```
<audio autoplay>
  <source src="audio.mp3" type="audio/mp3">
</audio>
```

#### 2. src:

- **Purpose:** This attribute specifies the location (URL) of the audio file you want to play. It is used to define the path to the audio file, so the browser knows which audio to load and play.

- **Example:** You provide the path or URL of the audio file you want to play.

```
<audio src="music.mp3">
</audio>
```

- **Note:** The `src` attribute can also be used inside the `<audio>` tag along with `<source>` tags if you want to specify multiple audio sources for better compatibility.

### 3. controls:

- **Purpose:** This attribute adds built-in controls to the audio player, such as play, pause, volume control, and the progress bar. It allows users to interact with the audio (e.g., play, pause, adjust volume) directly on the webpage.
- **Example:** The audio player will have buttons to control playback.

```
<audio controls>
  <source src="audio.mp3" type="audio/mp3">
</audio>
```



## 3. What is a URL? Explain the different parts of URL with example.

### What is a URL?

A **URL** (Uniform Resource Locator) is the address used to access a resource (like a webpage, image, or video) on the internet. It's essentially a web address that tells the browser where to find a specific resource.

### Parts of a URL:

#### 1. Protocol (Scheme):

- **Purpose:** Specifies the method used to access the resource on the internet. Common protocols include `http`, `https`, `ftp`, and `mailto`.
- **Example:** `https` is used for secure communication over the web.

#### 2. Domain Name:

- **Purpose:** This is the unique name of the website or server where the resource is hosted.
- **Example:** In the URL `https://www.example.com`, `www.example.com` is the domain name.

### 3. Port (Optional):

- **Purpose:** The port number defines a specific entry point to the server. By default, port `80` is used for HTTP and `443` for HTTPS. If a port is specified, it appears after the domain.
- **Example:** `https://www.example.com:8080` — here, `8080` is the port.

### 4. Path:

- **Purpose:** Specifies the location of a resource on the server (like a file or a webpage). This part of the URL identifies the exact resource.
- **Example:** In the URL `https://www.example.com/about`, `/about` is the path, referring to the "About" page on the website.

### 5. Query String (Optional):

- **Purpose:** Contains data that is passed to the server, usually in the form of key-value pairs. It starts with a `?` and can contain multiple parameters separated by `&`.
- **Example:** In the URL `https://www.example.com/search?query=hello&sort=desc`, `query=hello&sort=desc` is the query string, where `query` and `sort` are parameters.

### 6. Fragment (Optional):

- **Purpose:** A fragment identifier points to a specific section or element within a webpage. It starts with a `#` and allows jumping directly to that part of the page.
- **Example:** In the URL `https://www.example.com/page#section2`, `#section2` is the fragment identifier, which will take you to the "section2" part of the page.

## Example of a Full URL:

```
https://www.example.com:8080/products?id=1234&category=electronics#details
```

Here's the breakdown:

- **Protocol:** `https`
- **Domain Name:** `www.example.com`

- **Port:** 8080 (optional)
- **Path:** /products
- **Query String:** id=1234&category=electronics
- **Fragment:** #details



## 4. Describe World Wide Web and its significance.

The **World Wide Web (WWW)**, commonly known as the web, is a vast network of interconnected documents and resources, linked together by hyperlinks and accessible through the internet. It allows users to view and interact with websites, online services, and content, making it an essential part of everyday life for communication, entertainment, learning, and business.

### Key Components of the World Wide Web:

1. **Web Pages:** These are individual documents or content on the web, written in HTML (HyperText Markup Language). A web page can contain text, images, videos, links, and other multimedia.
2. **Websites:** A website is a collection of related web pages that are usually linked together under a common domain name (e.g., [www.example.com](http://www.example.com)).
3. **Web Browsers:** Programs like Google Chrome, Mozilla Firefox, and Safari that allow users to access and navigate the web by displaying web pages.
4. **Web Servers:** These are computers that store web pages, websites, and data, and serve them to users when requested via web browsers.
5. **URLs (Uniform Resource Locators):** These are addresses that identify and specify the location of web pages on the World Wide Web (e.g., <https://www.example.com>).

### Significance of the World Wide Web:

1. **Information Access:**
  - The WWW allows people to access a vast amount of information
2. **Communication:**
  - The web has revolutionized communication by enabling instant access to email, social media, and video conferencing platforms. People can stay connected with others worldwide through websites, apps, and services.

### 3. Business and Commerce:

- E-commerce (buying and selling goods and services online) has become a major part of the global economy.

### 4. Entertainment:

- The World Wide Web provides endless entertainment options.

### 5. Social Interaction:

- Social media platforms like Facebook, Twitter, Instagram, and LinkedIn are all part of the [WWW](#).



## *5. Differentiate between block tags and inline tags. Illustrate with suitable examples.*

### Block Tags

- **Purpose:** Block tags create sections or "blocks" of content.
- **Appearance:** These elements **start on a new line** and take up the **full width** of the container.
- **Behavior:** They stack like building blocks.
- **Examples:**
  - `<div>` : A general-purpose container for grouping elements.
  - `<p>` : Used for paragraphs of text.
  - `<h1>` , `<h2>` ... `<h6>` : Headings of different sizes.
  - `<ul>` : Unordered lists.
  - `<table>` : Creates a table.

### Example:

```
<h1>Heading</h1>
<p>This is a paragraph. It occupies its own block.</p>
<div>
  <p>Content inside a block.</p>
</div>
```

## Heading



This is a paragraph. It occupies its own block.

Content inside a block.



## Inline Tags

- **Purpose:** Inline tags format specific parts **within a line** of content.
- **Appearance:** They **don't start on a new line**; they stay in the flow of the text.
- **Behavior:** They are like puzzle pieces that fit within a block.
- **Examples:**
  - `<span>` : A general-purpose container for inline styling or scripts.
  - `<b>` : Makes text bold.
  - `<i>` : Italicizes text.
  - `<a>` : Creates hyperlinks.
  - `<img>` : Embeds an image.

### Example:

```
<p>This is a <b>bold</b> word, and <i>this</i> is italic.</p>  
<a href="https://example.com">Click here</a> to visit Example.com.
```

This is a **bold** word, and *this* is italic.

[Click here](#)

## Key Differences:

| Feature       | Block Tags  | Inline Tags   |
|---------------|---|---|
| Line Behavior | Starts on a new line  | Stays in the same line  |
| Width         | Takes up full width   | Only takes as much as needed  |
| Examples      | <code>&lt;div&gt;</code> , <code>&lt;p&gt;</code> , <code>&lt;h1&gt;</code> | <code>&lt;b&gt;</code> , <code>&lt;i&gt;</code> , <code>&lt;span&gt;</code> |

## Visual Analogy:

- **Block Tags:** Imagine a block tag as a paragraph in a book—it occupies a whole section.

- **Inline Tags:** Inline tags are like bold or italic words within that paragraph—they fit in without disrupting the flow.



## 6. Write the equivalent HTML code to implement the following in a web page:

- 1. An image titled “flower.jpg” with proper attribute to set height, width and message text.
- 2. Unordered list with values tea, coffee and milk

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Example</title>
</head>
<body>
  <!-- 1) Image with attributes -->
  
  <p>This is an image of a flower.</p>

  <!-- 2) Unordered list -->
  <ul>
    <li>Tea</li>
    <li>Coffee</li>
    <li>Milk</li>
  </ul>
</body>
</html>
```

### Explanation for the Exam:

1. The `<img>` tag is used to add an image to the web page.
  - `src`: Specifies the image source (file name or URL).
  - `alt`: Provides alternative text for the image.
  - `width` and `height`: Set the dimensions of the image.
2. The `<ul>` tag creates an unordered list.

- `<li>` : Used for each item in the list.



## 7. Explain following html tags with proper example.

- 1. `<textarea>` 2. `<span>` 3. `<tr>` 4. `<form>` 5. `<a>`

### 1. `<textarea>`

- **Purpose:** Used to create a multi-line text input area in a form.
- **Attributes:**
  - `rows` : Defines the number of visible rows.
  - `cols` : Defines the number of visible columns.
  - `placeholder` : Provides hint text.

**Example:**

```
<form>
  <label for="comments">Comments:</label>
  <textarea id="comments" rows="1" cols="10" placeholder="Write your
comments here..."></textarea>
</form>
```

Comments:

Write your comments here...

`\*\*

- **Purpose:** A general-purpose inline container used to style or group text without disrupting the flow.
- **Usage:** Commonly used with CSS or JavaScript for styling or scripting.

**Example:**

```
<p>This is a <span style="color: red;">highlighted</span> word.</p>
```

This is a **highlighted** word.

### 3. <tr>

- **Purpose:** Defines a table row inside a <table> element.
- **Usage:** Works with <td> (table data) and <th> (table header) to structure table content.

**Example:**

```
<table border="1">
  <tr>
    <th>Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>John</td>
    <td>25</td>
  </tr>
</table>
```

| Name | Age |
|------|-----|
| John | 25  |

```
### **4. `
`**
```

- **Purpose:** Creates a form for user input and data submission.
- **Attributes:**
  - **action:** URL where the form data is sent.
  - **method:** HTTP method (GET/POST).

**Example:**

```
<form action="/submit" method="post">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name">
  <input type="submit" value="Submit">
</form>
```

Name:

### \*\*5. ``\*\*

- **Purpose:** Creates a hyperlink to navigate to another page, section, or website.
- **Attributes:**
  - `href` : Specifies the link's destination.
  - `target` : Opens the link in a new tab if set to `_blank`.

### Example:

```
<a href="https://example.com" target="_blank">Visit Example</a>
```

[Visit Example](#)



## 8. Design a webpage that displays the following table.

| Company | Model        |
|---------|--------------|
| Dodge   | Challenger   |
| Maruti  | Swift        |
| Jeep    | Wrangler     |
| Ford    | Fusion       |
|         | Fiesta       |
|         | Escape       |
| BMW     | BMW 5 Series |

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Car Models</title>
</head>
<body>
  <table border="1">
    <thead>
      <tr>
        <th>Company</th>
        <th>Model</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>Dodge</td>
        <td>Challenger</td>
      </tr>
      <tr>
        <td>Maruti</td>
        <td>Swift</td>
      </tr>
      <tr>
        <td>Jeep</td>
        <td>Wrangler</td>
      </tr>
      <tr>
        <td data-kind="parent" data-rs="3">Ford</td>
        <td>Fusion</td>
      </tr>
      <tr>
        <td data-kind="ghost"></td>
        <td>Fiesta</td>
      </tr>
      <tr>
        <td data-kind="ghost"></td>
        <td>Escape</td>
      </tr>
      <tr>
        <td>BMW</td>
        <td>BMW 5 Series</td>
      </tr>
    </tbody>
  </table>

```

```

        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Dodge</td>
            <td>Challenger</td>
        </tr>
        <tr>
            <td>Maruti</td>
            <td>Swift</td>
        </tr>
        <tr>
            <td>Jeep</td>
            <td>Wrangler</td>
        </tr>
        <tr>
            <td rowspan="3">Ford</td>
            <td>Fusion</td>
        </tr>
        <tr>
            <td>Fiesta</td>
        </tr>
        <tr>
            <td>Escape</td>
        </tr>
        <tr>
            <td>BMW</td>
            <td>BMW 5 Series</td>
        </tr>
    </tbody>
</table>
</body>
</html>

```

## Explanation of the Table:

### 1. Headers:

- `<th>` tags are used for table headers "Company" and "Model".

### 2. Data Rows:

- Each car company and its models are represented using `<td>` inside `<tr>`.

### 3. Ford Models:

- The `rowspan="3"` attribute merges the "Ford" cell across three rows to display multiple models neatly.

### 4. Border:

- The `border="1"` attribute adds a border around the table for better visibility.

## Output

| Company | Model        |
|---------|--------------|
| Dodge   | Challenger   |
| Maruti  | Swift        |
| Jeep    | Wrangler     |
| Ford    | Fusion       |
|         | Fiesta       |
|         | Escape       |
| BMW     | BMW 5 Series |



**9. Write down the general format of an HTTP request and an HTTP response. What is the purpose of the following HTTP headers? Also, identify whether they are included with an HTTP header/response or both. i. host ii. last-modified**

### General Format of an HTTP Request

An HTTP request is how a client (usually a browser) sends a request to a server. Its general format is as follows:

```
<HTTP method> <Request-URI> <HTTP version>  
Headers: <Header-Name>: <Header-Value>  
Blank Line (End of Headers)  
[Optional Body for POST/PUT]
```

### Example of an HTTP Request

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html
```



## General Format of an HTTP Response

An HTTP response is what the server sends back to the client after processing the request. Its general format is:

```
<HTTP version> <Status-Code> <Reason-Phrase>
Headers: <Header-Name>: <Header-Value>
Blank Line (End of Headers)
[Optional Body]
```

## Example of an HTTP Response

```
HTTP/1.1 200 OK
Date: Mon, 25 Nov 2024 12:34:56 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 138

<html>
  <body>
    <h1>Welcome to Example!</h1>
  </body>
</html>
```



### i. Host

- **Purpose:**
  - Specifies the domain name of the server (and optionally the port number) being requested.



- Used in requests to identify the intended target server, especially in shared hosting environments where multiple websites reside on a single server.
- **Example:**

```
Host: www.example.com
```

- **Included With:** HTTP Request only.



## ii. Last-Modified

- **Purpose:**
  - Indicates the date and time when the requested resource was last modified.
  - Helps clients and caches determine if a resource has changed and avoid unnecessary data transfers.
- **Example:**

```
Last-Modified: Mon, 24 Nov 2024 12:34:56 GMT
```

- **Included With:** HTTP Response only.



## 10. How will you navigate between sections in the same web page? Illustrate with appropriate example?

- To navigate between sections on the same webpage, we use **anchor links** with the `id` attribute. This allows smooth or instant scrolling to specific sections of the page.
- **Assign an `id` to Each Section:** Each section you want to navigate to must have a unique `id`.
- **Create Links with `href="#id"`:** Use the `href` attribute in anchor tags ( `<a>` ) to link to the `id` of the target section.

### Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Simple Page Navigation</title>
</head>
<body>
  <!-- Navigation Links -->
  <nav>
    <a href="#section1">Go to Section 1</a> |
    <a href="#section2">Go to Section 2</a> |
    <a href="#section3">Go to Section 3</a>
  </nav>

  <!-- Sections -->
  <h1 id="section1">Section 1</h1>
  <p>This is the content of Section 1.</p>

  <h1 id="section2">Section 2</h1>
  <p>This is the content of Section 2.</p>

  <h1 id="section3">Section 3</h1>
  <p>This is the content of Section 3.</p>
</body>
</html>
```

## How It Works:

### 1. Links ( <a> ):

- Each <a> tag has an href pointing to the id of the section you want to navigate to.
- Example: <a href="#section1">Go to Section 1</a> navigates to the <h1> with id="section1".

### 2. Sections ( id ):

- Each <h1> element has a unique id that matches the href in the navigation links.

## Output in the Browser:

- Clicking "Go to Section 1" will jump to Section 1.

- Clicking "Go to Section 2" will jump to Section 2, and so on.



## 11. Create a HTML form for registering to a jobsite which includes fields to

- Enter your name
- address
- e-mail
- contact number
- a date picker to include your date of birth
- radio buttons to select you sex
- check boxes to show your area of interest
- a selection list to input your experience with a submit and reset button.
- All fields must be labelled appropriately

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Jobsite Registration Form</title>
</head>
<body>
  <h1>Jobsite Registration Form</h1>
  <form action="#" method="post">
    <!-- Name -->
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" required>
    <br><br>

    <!-- Address -->
    <label for="address">Address:</label>
    <textarea id="address" name="address" rows="4" cols="30" required>
  </textarea>
  <br><br>

  <!-- Email -->
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" required>
```

```

<br><br>

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

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

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

<!-- Areas of Interest -->
<label>Areas of Interest:</label>
<input type="checkbox" id="ai" name="interest" value="AI">
<label for="ai">Artificial Intelligence</label>
<input type="checkbox" id="webdev" name="interest" value="Web
Development">
<label for="webdev">Web Development</label>
<input type="checkbox" id="datasci" name="interest" value="Data
Science">
<label for="datasci">Data Science</label>
<br><br>

<!-- Experience -->
<label for="experience">Experience:</label>
<select id="experience" name="experience" required>
  <option value="" disabled selected>Select</option>
  <option value="0-1">0-1 years</option>
  <option value="1-3">1-3 years</option>
  <option value="3-5">3-5 years</option>

```

```
<option value="5+">5+ years</option>
</select>
<br><br>

<!-- Buttons -->
<button type="submit">Submit</button>
<button type="reset">Reset</button>
</form>
</body>
</html>
```

## Output


# Jobsite Registration Form

Name:

Address:

Email:

Contact Number:

Date of Birth:  

Gender: ☐ Male ☐ Female ☐ Other

Areas of Interest: ☐ Artificial Intelligence ☐ Web Development ☐ Data Science

Experience:  

### 1. Name, Address, Email, Contact Number:

- Text input and textarea fields, all with appropriate `label` tags and `required` attributes.

**2. Date Picker for Date of Birth:**

- `<input type="date">` for easy date selection.

**3. Radio Buttons for Gender:**

- Multiple options (Male, Female, Other), grouped under the `name="gender"`.

**4. Checkboxes for Areas of Interest:**

- Options like Artificial Intelligence, Web Development, etc.

**5. Dropdown List for Experience:**

- A `<select>` with predefined options.

**6. Submit and Reset Buttons:**

- `Submit` to send the data, and `Reset` to clear the form.

**12. Develop HTML code to create the following web page?**

**Image Searching Website**

---

Search your Image here:

[For more click here](#)

---

Feedback Form:

Name:

Date:

Comment:

```
<!DOCTYPE html>
<html>
<head>
  <title>Image Searching Website</title>
</head>
<body>
  <!-- Heading -->
  <h1 style="text-align:center">Image Searching Website</h1>
  <hr>
```

```
<!-- Search Section -->
Search your image here:
<input type="text" id="search-box" placeholder="Kung fu panda">
<button type="button">Search</button>
<br><br>

<!-- Hyperlink -->
<p>
  <a href="https://example.com" target="_blank">For more, click here</a>.
</p>
<hr>

<!-- Feedback Form -->
Feedback Form
<form action="#" method="post">
  <!-- Name -->
  <label for="name">Name:</label>
  <br>
  <input type="text" id="name" name="name" required>
  <br><br>

  <!-- Date -->
  <label for="date">Date:</label>
  <br>
  <input type="date" id="date" name="date" required>
  <br><br>

  <!-- Comment -->
  <label for="comment">Comment:</label>
  <br>
  <textarea id="comment" name="comment" rows="4" cols="30"></textarea>
  <br><br>

  <!-- Submit Button -->
  <button type="submit">Submit</button>
</form>
</body>
</html>
```

## Output

# Image Searching Website


Search your image here:

[For more, click here.](#)

Feedback Form

Name:

Date:

Comment:



***13. Give the syntax to create hyperlinks in HTML? List any 3 target attributes and their purpose associated with hyperlinks? Also how will you identify an active link, visited link and unvisited link?***

## Syntax to Create Hyperlinks in HTML

In HTML, hyperlinks are created using the `<a>` (anchor) tag. The basic syntax is:

```
<a href="URL">Link Text</a>
```

Where:

- `href` : The `href` attribute specifies the URL or the location to which the hyperlink points.
- `Link Text` : This is the clickable text that appears on the web page.

**Example:**



```
<a href="https://www.example.com">Visit Example</a>
```

## Target Attributes and Their Purpose

The `target` attribute specifies where to open the linked document. Here are three common `target` attributes:

### 1. `_blank`:

- **Purpose:** Opens the linked document in a new tab or window.
- **Example:**

```
<a href="https://www.example.com" target="_blank">Visit Example</a>
```

### 2. `_self` (default):

- **Purpose:** Opens the linked document in the same frame or window where the link was clicked. If no `target` is specified, this is the default behavior.
- **Example:**

```
<a href="https://www.example.com" target="_self">Visit Example</a>
```

### 3. `_parent`:

- **Purpose:** Opens the linked document in the parent frame (if the page is inside a `<frame>` or `<iframe>`).
- **Example:**

```
<a href="https://www.example.com" target="_parent">Visit Example</a>
```

### 4. `_top`:

- **Purpose:** Opens the linked document in the full body of the window, replacing any frames.
- **Example:**

```
<a href="https://www.example.com" target="_top">Visit Example</a>
```

# Identifying Active, Visited, and Unvisited Links

You can style links using CSS to distinguish between their different states. The states are:

## 1. Unvisited Link ( :link):

- **Purpose:** Targets links that have not been visited yet.
- **CSS Example:**

```
a:link {  
    color: blue; /* Color for unvisited links */  
}
```

## 2. Visited Link ( :visited):

- **Purpose:** Targets links that have been visited by the user.
- **CSS Example:**

```
a:visited {  
    color: purple; /* Color for visited links */  
}
```

## 3. Active Link ( :active):

- **Purpose:** Targets the link at the moment it is clicked (while the user is pressing the mouse button on it).
- **CSS Example:**

```
a:active {  
    color: red; /* Color for active links */  
}
```

## Example of Styling Links:

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>Link States Example</title>
<style>
  a:link {
    color: blue;
  }
  a:visited {
    color: purple;
  }
  a:active {
    color: red;
  }
</style>
</head>
<body>
  <a href="https://www.example.com">Visit Example</a>
</body>
</html>
```

In this example:

- The link will appear **blue** until it is clicked.
- After visiting the link, it will appear **purple**.
- While clicking the link, it will appear **red**.



## 14. Write the equivalent HTML code to implement the following in a web page:

- i. An image titled "flowers.jpg" with a height of 150 pixels and width of 250 pixels. If the image cannot be accessed, a message "No image available" should be displayed.
- ii. A hyperlink to the URL "[www.mysite.com/birds.jpg](http://www.mysite.com/birds.jpg)". The hyperlink should have the label "Click Here".
- iii. An unordered list with values Tea, Coffee, Milk.

### i. Flowers.jpg image

To display an image titled "flowers.jpg" with a specified height and width, and show a fallback message if the image cannot be accessed, use the `alt` attribute.

```

```

- `src="flowers.jpg"` : The image source.
- `alt="No image available"` : Text displayed if the image cannot be loaded.
- `height="150"` and `width="250"` : The specified dimensions of the image.

## ii. Hyperlink to "birds.jpg"

To create a hyperlink to the URL "[www.mysite.com/birds.jpg](http://www.mysite.com/birds.jpg)" with the label "Click Here", use the `<a>` tag.

```
<a href="http://www.mysite.com/birds.jpg">Click Here</a>
```

- `href="http://www.mysite.com/birds.jpg"` : The destination URL for the hyperlink.
- The text between the opening and closing `<a>` tags is the link label ("Click Here").

## iii. Unordered List with Tea, Coffee, and Milk

To create an unordered list with the items "Tea," "Coffee," and "Milk," use the `<ul>` (unordered list) and `<li>` (list item) tags.

```
<ul>
  <li>Tea</li>
  <li>Coffee</li>
  <li>Milk</li>
</ul>
```

- `<ul>` : The tag for an unordered list.
- `<li>` : The tag for each list item.

## Complete HTML Code:

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

```

<title>HTML Example</title>
</head>
<body>

  <!-- Image with fallback text -->
  

  <!-- Hyperlink to birds.jpg -->
  <a href="http://www.mysite.com/birds.jpg">Click Here</a>

  <!-- Unordered list -->
  <ul>
    <li>Tea</li>
    <li>Coffee</li>
    <li>Milk</li>
  </ul>

</body>
</html>

```

This code implements:

- An image with fallback text if not available.
- A hyperlink to another image file.
- An unordered list of drinks.



## 15. Write the HTML code for obtaining the following table.

| Flower Show |        |
|-------------|--------|
| Flower Name | Colour |
| Red Rose    | Red    |
| White Rose  | White  |
| Jasmine     |        |
| Sunflower   | Yellow |

```

<table>
  <tr>
    <td colspan="2">Flower Show</td>
  </tr>
  <tr>
    <th>Flowername</th>
    <th>Colour</th>
  </tr>
  <tr>
    <td>Red Rose</td>
    <td>Red</td>
  </tr>
  <tr>
    <td>White Rose</td>
    <td rowspan="2">White</td>
  </tr>
  <tr>
    <td>Jasmin</td>
  </tr>
  <tr>
    <td>Sunflower</td>
    <td>Yellow</td>
  </tr>
</table>

```

| Flower Show |        |
|-------------|--------|
| Flowername  | Colour |
| Red Rose    | Red    |
| White Rose  | White  |
| Jasmin      |        |
| Sunflower   | Yellow |



## 16. Explain list tags with example.

### 1. Ordered List (< ol>)

An ordered list is a list in which the items are numbered or alphabetized. Each item in the list is defined by the `<li>` (list item) tag.

### Example of an Ordered List:

```
<ol>
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
</ol>
```

### Output:

1. First item
2. Second item
3. Third item

You can also specify the type of numbering (numbers, letters, roman numerals) using the `type` attribute:

```
<ol type="a">
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
</ol>
```

This will result in:

- a. First item
- b. Second item
- c. Third item

## 2. Unordered List (< ul>)

An unordered list is a list where the items are not in any particular order. Typically, items in an unordered list are shown with bullet points. Again, each item in the list is defined by the `<li>` tag.

### Example of an Unordered List:

```
<ul>
  <li>Apple</li>
  <li>Banana</li>
  <li>Cherry</li>
</ul>
```

### Output:

- Apple
- Banana
- Cherry



**17. Write the HTML code for obtaining the following registration form, where the course field contains the options BCA, BBA, B. Tech, MBA, MCA, M. Tech**

Firstname

Middlename:

Lastname:

Course :

Gender :

☐ Male

☐ Female

☐ Other

Phone :

Address

Email:

Password:

Re-type password:



```

<!DOCTYPE html>
<html>
<head>
  <title>Simple Registration Form</title>
</head>
<body>

  <form>
    First Name: <input type="text" name="first-name" required><br><br>
    Middle Name: <input type="text" name="middle-name"><br><br>
    Last Name: <input type="text" name="last-name" required><br><br>
    Course:
    <select name="course" placeholder="Course">
      <option value="" disabled selected>Course</option>
      <option value="BCA">BCA</option>
      <option value="BBA">BBA</option>
      <option value="Btech">BTech</option>
      <option value="MBA">MBA</option>
      <option value="MCA">MCA</option>
      <option value="MTech">MTech</option>
    </select><br><br>
    Gender: <br>
    <input type="radio" name="gender" value="Male" required> Male<br>
    <input type="radio" name="gender" value="Female"> Female<br>
    <input type="radio" name="gender" value="Other"> Other<br><br>
    Phone:
    <input type="text" name="country-code" placeholder="+91"
style="width: 50px;" required>
    <input type="text" name="phone-number" required><br><br>
    Address: <br>
    <textarea name="address" rows="4" cols="40" required></textarea><br>
  <br>
    Email: <input type="email" name="email" required><br><br>
    Password: <input type="password" name="password" required><br><br>
    Re-type Password: <input type="password" name="retype-password"
required><br><br>
    <button type="submit">Submit</button>
  </form>

</body>
</html>

```

## Output

First Name:

Middle Name:

Last Name:

Course:

Course ▾

Gender:

☐ Male

☐ Female

☐ Other

Phone:

+91

Address:

Email:

Password:

Re-type Password:

Submit



## ***18. Explain HTTP and its significance. Describe the request and response phases in HTTP.***

HTTP (HyperText Transfer Protocol) is the foundational protocol used for communication between clients (like web browsers) and servers over the internet. It defines the rules for requesting and receiving resources, such as HTML documents, images, and other web content. HTTP allows the web to function by enabling the transfer of data between clients and servers.

### **Significance of HTTP:**

1. **Web Communication:** HTTP is the protocol that enables browsers to request and display web pages from servers.
2. **Stateless:** HTTP is a stateless protocol, meaning each request is independent and does not retain information about previous requests, providing simplicity and scalability.
3. **Foundation for Web Services:** It powers all web services, including APIs, websites, and cloud applications.
4. **Compatibility:** HTTP works across different operating systems and devices, making it universally compatible on the web.
5. **Security:** When combined with SSL/TLS (making it HTTPS), HTTP ensures secure communication over the internet by encrypting data during transmission.

## HTTP Request and Response Phases

### 1. HTTP Request Phase

The request phase begins when the client (usually a web browser or an app) sends a request to the server. This is how the client asks for resources or services from the server.

#### Components of an HTTP Request:

- **Request Line:**
  - **Method:** The action to be performed (e.g., GET, POST, PUT, DELETE).
  - **URL:** The resource the client is requesting (e.g., `/home`, `/api/data`).
  - **HTTP Version:** The version of HTTP being used (e.g., `HTTP/1.1`). Example: `GET /index.html HTTP/1.1`
- **Headers:** Provide additional information about the request. They describe the client, what kind of data is being sent or accepted, and more. Example headers: `User-Agent`, `Accept`, `Authorization`, `Content-Type`.
- **Body (Optional):** The body contains data being sent to the server, often used in methods like POST or PUT (e.g., form submissions, JSON data for APIs). Example: JSON data for submitting a form or an API request.

### 2. HTTP Response Phase

Once the server receives the request, it processes the request and returns a response.

#### Components of an HTTP Response:

- **Status Line:**
  - **HTTP Version:** The version of HTTP being used (e.g., `HTTP/1.1`).
  - **Status Code:** A 3-digit code indicating the outcome of the request (e.g., `200` for success, `404` for not found).
  - **Reason Phrase:** A short description of the status code (e.g., `OK`, `Not Found` ).  
Example: `HTTP/1.1 200 OK`
- **Headers:** Metadata about the response, such as content type, cache settings, or authentication info. Example headers: `Content-Type`, `Cache-Control`, `Location` (for redirects).
- **Body (Optional):** Contains the content being returned to the client. This could be HTML, JSON, an image, or any other type of data requested by the client. Example: The HTML of a web page, JSON data from an API, or an image file.



## ***19. Explain MIME and its types with examples. Describe why should MIME type information be essentially included in HTTP responses.***

### **MIME (Multipurpose Internet Mail Extensions) and Its Significance**

- MIME is a standard that extends the original format of emails to allow the inclusion of various types of content (such as text, images, audio, and video) beyond the traditional text.
- Originally developed for email, MIME has since been adapted for use in HTTP (HyperText Transfer Protocol) responses to define the type of content being sent between the client and the server.
- MIME types (also known as Content-Type) are **used to specify the nature or format of a document, file, or data being transferred over the internet**, ensuring that the recipient can interpret and display the content properly.

### **Types of MIME Types**

MIME types are divided into two main parts:

1. **Type:** This indicates the general category of the content (e.g., text, image, audio).

2. **Subtype**: This specifies the exact format of the content (e.g., HTML, JPEG, JSON).

Here are the main categories and some common subtypes of MIME types:

## 1. Text

- **text/plain** : Plain text, no formatting.

Example: `Hello, world!`

- **text/html** : HTML content.

Example: `<html><body><h1>Welcome to My Website</h1></body></html>`

- **text/css** : CSS stylesheets.

Example: `body { color: red; }`

- **text/javascript** or **application/javascript** : JavaScript files.

Example: `console.log('Hello, world!');`

## 2. Image

- **image/jpeg** : JPEG images.

Example: A `.jpg` file, commonly used for photographs.

- **image/png** : PNG images.

Example: A `.png` file, used for images with transparency.

- **image/gif** : GIF images.

Example: A `.gif` file, often used for animated images.

## 3. Audio/Video

- **audio/mpeg** : MP3 audio files.

Example: A `.mp3` file, commonly used for music.

- **video/mp4** : MP4 video files.

Example: A `.mp4` file, widely used for video streaming.

- **audio/ogg** : Ogg Vorbis audio files.

Example: A `.ogg` file, an open-source audio format.

## 4. Application

- **application/json** : JSON data.

Example: `{ "name": "John", "age": 30 }`

- **application/xml** : XML data.

Example: `<person><name>John</name><age>30</age></person>`

- **application/pdf** : PDF documents.

Example: A `.pdf` file, used for documents that preserve formatting.

- **application/zip** : Zip archive files.

Example: A `.zip` file, commonly used for compressing multiple files.

## 5. Multipart

- **multipart/form-data** : Used for submitting forms that include files, such as when uploading an image or document.
- **multipart/alternative** : Used for emails containing both plain text and HTML content, allowing the recipient to choose which format to view.

## Why MIME Type Information Should Be Included in HTTP Responses

MIME types are essential for correct interpretation of data sent from a server to a client. Including MIME type information in HTTP responses has several important benefits:

### 1. Correct Content Rendering

The MIME type informs the client (browser, app, etc.) how to process the received data. For example:

- If the MIME type is `text/html`, the browser knows to interpret the data as an HTML page.
- If the MIME type is `application/json`, the client knows it should parse the data as JSON.

Without MIME type information, the client may not know how to correctly interpret the content, leading to errors or improper rendering of the data.

### 2. Content-Type Negotiation

MIME types allow content negotiation between the client and server. The client can request a specific type of content using the `Accept` header, and the server can respond with the appropriate MIME type.

- For example, a browser might request an image in PNG format ( `Accept: image/png` ), and the server would then send the image in that format if available.

### 3. Proper Handling of Different Data Formats

MIME types allow a server to send multiple data formats for different kinds of requests:

- APIs often use `application/json` to return data in JSON format, which is easily processed by JavaScript.
- When delivering files like images or PDFs, the correct MIME type ensures the file is handled properly by the client.

### 4. Efficient File Transfers

Including MIME types can help clients identify content before downloading it fully. For example, a client might choose not to download an image in an unsupported format if the MIME type indicates it is not the format it expects.



### Example of HTTP Response with MIME Type:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 123

{
  "name": "John",
  "age": 30
}
```

In this example

- The `Content-Type: application/json` header tells the client that the response body contains JSON data.

- The client will process the response as JSON, parsing it into an object for use within the application.



**20. Write HTML code to design a web page to create a table with 5 rows and 3 columns for entering the mark list of 4 students. Assume suitable headings for each column.**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Student Mark List</title>
</head>
<body>

  <h2>Student Mark List</h2>

  <table border="1">
    <thead>
      <tr>
        <th>Student Name</th>
        <th>Subject</th>
        <th>Marks</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td><input type="text" value="John Doe"></td>
        <td><input type="text" value="Mathematics"></td>
        <td><input type="text" value="85"></td>
      </tr>
      <tr>
        <td><input type="text" value="Alice Smith"></td>
        <td><input type="text" value="Science"></td>
        <td><input type="text" value="90"></td>
      </tr>
      <tr>
```



```

        <td><input type="text" value="Bob Johnson"></td>
        <td><input type="text" value="History"></td>
        <td><input type="text" value="78"></td>
    </tr>
    <tr>
        <td><input type="text" value="Catherine Lee"></td>
        <td><input type="text" value="English"></td>
        <td><input type="text" value="92"></td>
    </tr>
    <tr>
        <td><input type="text" value="David Brown"></td>
        <td><input type="text" value="Computer Science"></td>
        <td><input type="text" value="88"></td>
    </tr>
</tbody>
</table>

</body>
</html>

```

| Student Name  | Subject          | Marks |
|---------------|------------------|-------|
| John Doe      | Mathematics      | 85    |
| Alice Smith   | Science          | 90    |
| Bob Johnson   | History          | 78    |
| Catherine Lee | English          | 92    |
| David Brown   | Computer Science | 88    |