Q1 a. Define HTTP.Explain the different Phases of HTTP.

Answer:

THE HYPERTEXT TRANSFER PROTOCOL

• HTTP contains two phases request phase and response phase.

• Each HTTP communication between browser and server consist of two part, a header

and a body, header contain information about communication and body contain data or

message of the communication if there is any.

Request Phase:

The general form of an HTTP request is as follows:

1. HTTP method Domain part of the URL HTTP version

2. Header fields

3. Blank line

4. Message body

The following is an example of the first line of an HTTP request:

GET /storefront.html HTTP/1.1

The format of a header field is the field name followed by a colon and the value of the field.

There are four categories of header fields:

1. General: For general information, such as the date

2. Request: Included in request headers

3. Response: For response headers

4. Entity: Used in both request and response headers

A wildcard character, the asterisk (*), can be used to specify that part of a MIME type can be

Anything.

The Host: host name request field gives the name of the host. The Host field is required for

HTTP 1.1. The If-Modified-Since: date request field specifies that the requested file should be

sent only if it has been modified since the given date. If the request has a body, the length of

that body must be given with a Content-length field. The header of a request must be followed

by a blank line, which is used to separate the header from the body of the request.

The Response Phase:

The general form of an HTTP response is as follows:

1. Status line

2. Response header fields

3. Blank line

4. Response body

The status line includes the HTTP version used, a three-digit status code for the response, and a

short textual explanation of the status code.

For example, most responses begin with the following:

HTTP/1.1 200 OK

The status codes begin with 1, 2, 3, 4, or 5. The general meanings of the five categories

specified by these first digits are shown in Table 1.2.

One of the more common status codes is one user never want to see: 404 Not Found, which

means the requested file could not be found.

b. Briefly explain the following:

1.URL 2.MIME 3.web server 4.web browser

MULTIPURPOSE INTERNET MAIL EXTENSIONS

• MIME stands for Multipurpose Internet Mail Extension.

• The server system apart from sending the requested document, it will also send MIME

information.

• The MIME information is used by web browser for rendering the document properly.

• The format of MIME is: type/subtype

• Example: text/html , text/doc , image/jpeg , video/mpeg

• When the type is either text or image, the browser renders the document without any

problem

• However, if the type is video or audio, it cannot render the document

• It has to take the help of other software like media player, win amp etc.,

• These softwares are called as helper applications or plugins

• These non-textual information are known as HYPER MEDIA

• Experimental document types are used when user wants to create a customized information & make it available in the internet

• The format of experimental document type is: type/x-subtype

• Example: database/x-xbase , video/x-msvideo

• Along with creating customized information, the user should also create helper applications.

• This helper application will be used for rendering the document by browser.

• The list of MIME specifications is stored in configuration file of web server.

UNIFORM RESOURCE LOCATORS

• Uniform Resource Locators (URLs) are used to identify different kinds of resources on

Internet.

• If the web browser wants some document from web server, just giving domain name is

not sufficient because domain name can only be used for locating the server.

• It does not have information about which document client needs. Therefore, URL should

be provided.

• The general format of URL is: scheme: object-address

• Example: http: www.vtu.ac.in/results.php

• The scheme indicates protocols being used. (http, ftp, telnet...)

• In case of http, the full form of the object address of a URL is as follows:

//fully-qualified-domain-name/path-to-document

• URLs can never have embedded spaces

• It cannot use special characters like semicolons, ampersands and colons

• The path to the document for http protocol is a sequence of directory names and a

filename, all separated by whatever special character the OS uses. (Forward or

backward slashes)

• The path in a URL can differ from a path to a file because a URL need not include all

directories on the path

• A path that includes all directories along the way is called a complete path

• Example: http://www.gumboco.com/files/f99/storefront.html

• In most cases, the path to the document is relative to some base path that is specified in

the configuration files of the server. Such paths are called partial paths.

• Example: http://www.gumboco.com/storefront.html

WEB SERVERS

Web servers are programs that provide documents to requesting browsers.
Example: Apache

Web server operations:

• All the communications between a web client and a web server use the HTTP

• When a web server begins execution, it informs the OS under which it is running & it

runs as a background process

• A web client or browser, opens a network connection to a web server, sends

information requests and possibly data to the server, receives information from the

server and closes the connection.

• The primary task of web server is to monitor a communication port on host machine,

accept HTTP commands through that port and perform the operations specified by the

commands.

• When the URL is received, it is translated into either a filename or a program name

General characteristics of web server:

• The file structure of a web server has two separate directories

• The root of one of these is called document root which stores web documents

• The root of the other directory is called the server root which stores server and its

support software's

• The files stored directly in the document root are those available to clients through top

level URLs

• The secondary areas from which documents can be served are called virtual document

trees.

• Many servers can support more than one site on a computer, potentially reducing the

cost of each site and making their maintenance more convenient. Such secondary hosts

are called virtual hosts.

• Some servers can serve documents that are in the document root of other machines on

the web; in this case they are called as proxy servers.

WEB BROWSERS

• Documents provided by servers on the Web are requested by browsers, which are

programs running on client machines.

• They are called browsers because they allow the user to browse the resources available

on servers.

• Mosaic was the first browser with a graphical user interface.

• A browser is a client on the Web because it initiates the communication with a server,

which waits for a request from the client before doing anything.

• In the simplest case, a browser requests a static document from a server.

• The server locates the document among its servable documents and sends it to the

browser, which displays it for the user.

• Sometimes a browser directly requests the execution of a program stored on the server.

• The output of the program is then returned to the browser.

• Examples: Internet Explorer, Mozilla Firefox, Netscape Navigator, Google Chrome, Opera

etc.

Q 2.a Discuss the basic structure of aHTML5 webpage.

Answer:

Standard XHTML Document Structure

• Every XHTML document must begin with an xml declaration element that simply identifies the document as being one based on XML. This element includes an attribute that specifies the version number 1.0.

• The xml declaration usually includes a second attribute, encoding, which specifies the encoding used for the document, utf-8

• Following is the xml declaration element, which should be the first line of every XHTML document:

```
<?xml version = "1.0" encoding = "utf-8"?>
```

• Note that this declaration must begin in the first character position of the document file

• The xml declaration element is followed immediately by an SGML DOCTYPE command which specifies the particular SGML document-type definition (DTD) with which the document complies, among other things.

• The following command states that the document in which it is included complies

with XHTML 1.0

□!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">□

• An XHTML document must include the four tags <html>, <head>, <title>, and

<body>.

• The <html> tag identifies the root element of the document So, XHTML documents

always have an <html> tag immediately following the DOCTYPE command, and

they always end with the closing html tag, </html>.

• The html element includes an attribute, xmlns, that specifies the XHTML namespace,

as shown in the following element:

<html xmlns = "http://www.w3.org/1999/xhtml">

• Although the xmlns attribute's value looks like a URL, it does not specify a

document. It is just a name that happens to have the form of a URL.

• An XHTML document consists of two parts, named the head and the body.

• The <head> element contains the head part of the document, which provides

information about the document and does not provide the content of the document.

• The body of a document provides the content of the document.

• The content of the title element is displayed by the browser at the top of its display

window, usually in the browser window ’s title bar

b.List and explain any three from elements in HTML5 with a suitable example.

1.<video> Element:

- Explanation: The **<video>** element is used to embed videos on a web page. It allows you to display video content and control playback using various attributes and methods.
- Example:

```
<video width="320" height="240" controls>
  <source src="example.mp4" type="video/mp4">
  Your browser does not support the video tag.
</video>
```

In this example, a video is embedded with a specified width and height, and the "controls" attribute provides playback controls like play, pause, and volume.

2.<canvas> Element:

- Explanation: The **<canvas>** element provides a drawing surface for creating graphics and animations using JavaScript. You can draw shapes, images, and more on the canvas.
- Example:

```
<canvas id="myCanvas" width="400" height="200"></canvas>
<script>
  var canvas = document.getElementById("myCanvas");
  var ctx = canvas.getContext("2d");
  ctx.fillStyle = "blue";
  ctx.fillRect(50, 50, 100, 100);
</script>
```

3. *<input> Element (with type="email"):*

- *Explanation: The <input> element is used for various form controls, and when used with the "type" attribute set to "email," it is used to input email addresses. It provides built-in validation for email input.*
- *Example:*

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

In this example, an email input field is created, and the "required" attribute ensures that the user must enter a valid email address before submitting the form

c.Explain the following tags with examples

1. Heading tag 2.Hypertext link tag 3.Image tag

4. Audio and vedio tag 5. Progress tag

Ans:

Heading tag

In XHTML, there are six levels of headings, specified by the tags <h1>, <h2>, <h3>, <h4>, <h5>, and <h6>, where <h1> specifies the highest-level heading. Headings are usually displayed in a boldface font whose default size depends on the number in the heading tag. On most browsers, <h1>, <h2>, and <h3> use font sizes that are larger than that of the default size of text, <h4> uses the default size, and <h5> and <h6> use smaller sizes. The heading tags always break the current line, so their content always appears on a new line. Browsers usually insert some vertical space before and after all headings

<html>

<head><title> Headings </title></head>

<body>

<h1> Heading 1 </h1>

<h2> Heading 2 </h2>

<h3> Heading 3 </h3>

<h4> Heading 4 </h4>

<h5> Heading 5 </h5>

<h6> Heading 6 </h6>

</body>

</html>

(ii) Hypertext link tag

The hypertext link tag in HTML is represented by the <a> element, which stands for "anchor." It is one of the fundamental elements used to create hyperlinks on web pages. Hyperlinks are clickable elements that allow users to navigate from one webpage to another, view a different section of the same webpage, download files, or link to other resources on the internet.

Here is the basic syntax of the <a> element:

Link Text

- href Attribute: The href attribute specifies the destination URL (Uniform Resource Locator) where the link points. It can be a web address, a file path, or an email address.
- Link Text: This is the visible text or content of the hyperlink that users see on the webpage. Clicking on this text activates the link.

Examples:

1. Creating a Basic Web Link:

Visit Example Website

In this example, clicking "Visit Example Website" will take the user to the https://www.example.com website.

2. Linking to a Different Section on the Same Page (Internal Link):

Go to Section

Here, #section-id refers to the ID attribute of the HTML element you want to link to. For example, if you have a section with the ID attribute set to section-id, clicking "Go to Section" will scroll the user to that specific section on the same page.

3. Linking to an Email Address:

Email Us

This creates a link that opens the user's default email client with the recipient address set to example@example.com when clicked.

4. Linking to a File (e.g., PDF, Image):

Download PDF

Clicking "Download PDF" will prompt the user to download the document.pdf file from the specified location on the server.

5. Linking to a Different Web Page in the Same Website:

Go to Page 2

This creates a link to another page within the same website. /page2.html represents the relative path to the target page from the root directory of the website.

The <a> element is incredibly versatile and is a fundamental building block for navigation and interaction on the web. It allows for seamless connectivity between different web resources, providing users with a smooth browsing experience.

(iii) Image tag

The image tag in HTML is represented by the <img> element. It is used to embed images in a web page. Images enhance the visual appeal of a webpage and are essential for creating engaging and interactive content.

Here is the basic syntax of the <img> element:

<img src="URL or file path" alt="Description of the image" width="width_value" height="height_value">

- src Attribute: The src attribute specifies the source URL or file path of the image. It can be a web address (URL) or a local file path pointing to the image file on the server or device.
- alt Attribute (Optional): The alt attribute provides alternative text for the image. It is displayed if the image cannot be loaded and is also used by screen readers for accessibility. It should describe the content or purpose of the image.
- width and height Attributes (Optional): These attributes define the width and height of the image in pixels. They are optional, and if not specified, the browser will render the image in its original size.

Example:

<img src="example.jpg" alt="Sample Image" width="300" height="200">

In this example, the <img> element displays an image named example.jpg. If the image is not found, the text "Sample Image" will be displayed instead. Additionally, the image will be displayed with a width of 300 pixels and a height of 200 pixels.

(iv) Audio and video tags

Certainly! Here are explanations and examples for three common HTML5 elements:

1. <header> Element:

The <header> element represents a container for introductory content or a set of navigational links in a document. It typically contains headings, logos, and navigation menus.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Website Header Example</title>
</head>
<body>
  <header>
    <h1>My Website</h1>
    <nav>
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">About</a></li>
        <li><a href="#">Services</a></li>
        <li><a href="#">Contact</a></li>
      </ul>
    </nav>
  </header>

  <h2>Main Content Goes Here</h2>
  <!-- Rest of the webpage content -->
```

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

In this example, the <header> element contains the website's title and a navigation menu.

2. <section> Element:

The <section> element defines a section in a document. It is often used to group related content together and can be thought of as a thematic grouping within a webpage.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Section Example</title>
</head>
<body>
  <section>
    <h2>Section 1</h2>
    <p>This is the first section of the document.</p>
  </section>

  <section>
    <h2>Section 2</h2>
    <p>This is the second section of the document.</p>
```

```
    </section>

    <!-- More sections can be added here -->

</body>
</html>
```

In this example, two <section> elements group related content. Each section has a heading and a paragraph of text.

3. <input> Element:

The <input> element is used to create interactive controls within web forms. It allows the user to enter data, which can include various types such as text, numbers, dates, checkboxes, and radio buttons.

Example:

```
<!DOCTYPE html>
<html>
<head>
    <title>Input Element Example</title>
</head>
<body>
    <h2>User Registration</h2>
    <form>
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required><br>

        <label for="email">Email:</label>
```

```
    <input type="email" id="email" name="email" required><br>

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

    <input type="submit" value="Register">
  </form>
</body>
</html>
```

In this example, the <input> element is used to create fields for entering a username, email, and password within a user registration form. The type attribute specifies the type of input field, and the required attribute ensures that the user must fill out these fields before submitting the form.

5.Progress bar

The `<progress>` tag represents the completion progress of a task.

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

Use the `<progress>` tag in conjunction with JavaScript to display the progress of a task.

The `<progress>` tag is not suitable for representing a gauge (e.g. disk space usage or relevance of a query result). To represent a gauge, use the <meter> tag instead.

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

```
<h1>The progress element</h1>

<label for="file">Downloading progress:</label>
<progress id="file" value="32" max="100"> 32% </progress>

</body>
</html>
```

3. what is CSS? Describe the different level of CSS style sheet and there precedence?

XHTML style sheets are called cascading style sheets because they can be defined at three different levels to specify the style of a document. Lower level style sheets can override higher level style sheets, so the style of the content of a tag is determined, in effect, through a cascade of style-sheet applications..

CSS, which stands for Cascading Style Sheets, is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. It controls the layout, appearance, and formatting of web pages, enabling developers to separate content from design and layout. CSS allows developers to create consistent styles across multiple pages, which can simplify the process of managing and updating a website.

LEVELS OF STYLE SHEETS

- The three levels of style sheets, in order from lowest level to highest level, are inline, document level, and external.
- **Inline style sheets** apply to the content of a single XHTML element.
- **Document-level style sheets** apply to the whole body of a document.
- **External style sheets** can apply to the bodies of any number of documents.
- Inline style sheets have precedence over document style sheets, which have precedence over external style sheets.
- Inline style specifications appear within the opening tag and apply only to the content of that tag.
- Document-level style specifications appear in the document head section and apply to the entire body of the document.
- External style sheets stored separately and are referenced in all documents that use them.
- External style sheets are written as text files with the MIME type text/css. They can be stored on any computer on the Web. The browser fetches external style sheets just as it fetches documents.
- The <link>tag is used to specify external style sheets. Within <link>, the rel attribute is used to specify the relationship of the linked-to document to the document in which the link appears. The href attribute of <link>is used to specify the URL of the style sheet document.

STYLE SPECIFICATION FORMATS

The format of style specification depends on the level of stylesheet.

Inline Style Specification: appears as values of the style attribute of a tag, the general form is as follows:

Style = "Property1 : Value1; Property2 : Value2; Property3 : Value3; ..... Property_n:Value_n;"

It is recommended that last property/value pair be followed by a semicolon.

Eg:

```
<h1 style ="font-family: 'Lucida Handwriting'; font-size: 50pt; color: Red;">Web Technology</h1>
```

Document Style Specification: appears as the content of a style element within the header of a document, general form of the content of a style element is as follows:

```
<style type = "text/css">
        Rule list
</style>
```

Each style rule in a rule list has two parts: a selector, which indicates the tag or tags affected by the rule, and a list of property–value pairs. The list has the same form as the quoted list for inline style sheets, except that it is delimited by braces rather than double quotes. So, the form of a style rule is as follows:

Selector { *Property1 : Value1; Property2 : Value2; Property3 : Value3; .....*

***Property_n:Value_n;* }**

Eg:

```
<style type = "text/css">
        h1 {
                font-family: 'Lucida Handwriting';
                font-size: 50pt;
                color: Red;
        }
</style>
```

External Style Sheet: have a form similar to that of document style sheets. The external file consists of a list of style rules.

Eg

```
<head>
        <link rel="stylesheet" type="text/css" href="cssfile.css">
</head>
```

Cssfile.css

```
P{
        Background-color:blue;
        Color:red;
}
```

CSS, which stands for Cascading Style Sheets, is a language used for describing the presentation of a document written in HTML or XML (including XML dialects like SVG or XHTML). It defines how elements of a web page should be displayed, such as layout, colors, fonts, and other design aspects. CSS separates the content of a web page from its visual presentation, allowing web developers to control the appearance of web pages efficiently.

CSS styles can be applied at different levels, and they have a specific order of precedence when multiple styles conflict. These levels of CSS styles are:

1. **Inline Styles:** These styles are applied directly to individual HTML elements using the `style` attribute. They have the highest specificity and will override any other styles. For example:

```html
<p style="color: red; font-size: 16px;">This is a red and 16px text.</p>
```

2. **Internal Styles (Embedded Styles):** These styles are defined within the `<style>` element in the HTML document's `<head>` section. They apply to all elements on that specific page. They have a medium level of specificity. For example:

```html
<head>
  <style>
    p {
      color: blue;
      font-size: 18px;
    }
  </style>
</head>
<body>
  <p>This is a blue and 18px text.</p>
</body>
```

3. **External Styles (External Style Sheets):** These styles are defined in separate .css files and linked to HTML documents. They have the lowest specificity and can

be applied to multiple web pages. When there are conflicting styles, the last rule applied takes precedence. For example:

CSS in a file named `styles.css`:

```css
p {
  color: green;
}
```

HTML file linking to the external stylesheet:

```html
<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <p>This is a green text.</p>
</body>
```

When there are conflicting styles between these levels, the order of precedence (from highest to lowest) is as follows:

1. **Inline Styles**: These override all other styles.

2. **Internal Styles (Embedded Styles)**: These will override external styles but can be overridden by inline styles.

3. **External Styles (External Style Sheets)**: These are the least specific and will be overridden by both inline and internal styles.

In addition to these levels, CSS also employs the concept of specificity and the order of declaration in the stylesheet to determine which styles should apply when there are conflicts. Specificity is a complex topic that assigns values to different selectors, allowing the browser to determine which style to use when multiple styles conflict.

SELECTOR FORMS

1) **Simple Selector Forms:**

In case of simple selector, a tag is used. If the properties of the tag are changed, then it reflects at all the places when used in the program. The selector can be any tag. If the new properties for a tag are not mentioned within the rule list, then the browser uses default behaviour of a tag.

Eg:

```
h1 { font-size : 24pt; }

h2, h3{ font-size : 20pt; }

body b em { font-size : 14pt; }
```

Only applies to the content of 'em' elements that are descendent of bold element in the body of the document. This is a **contextual selector**

2) Class Selectors:

Class selectors are used to allow different occurrences of the same tag to use different style specifications.

Eg

```
<head>
    <style type = "text/css">
        p.one { font-family: 'Lucida Handwriting'; font-size: 25pt; color: Red; }
        p.two{ font-family: 'Monotype Corsiva'; font-size: 50pt; color: green; }
    </style>
</head>
<body>
    <p class = "one">Web Technology</p>
    <p class = "two">Web Technology</p>
</body>
```

3) Generic Selectors:

Sometimes it is convenient to have a class of Style specification that applies to the content of more than one kind of tag. This is done by using a generic class, which is defined without a tag name in its name. In place of the tag name, you use the name of the generic class, which must begin with a period.

Eg

```
<head>
        <style type = "text/css">
                .sale{ font-family: 'Monotype Corsiva'; color: green; }
        </style>

</head>

<body>

        <p class = "sale">Weekend Sale</p>

        <h1 class = "sale">Weekend Sale</h1>

        <h6 class = "sale"> Weekend Sale</h6>

</body>
```

4) **id Selectors:**

An id selector allows the application of a style to one specific element.

Eg:

```
<head>
    <style type = "text/css">
        #one { font-family: 'Lucida Handwriting'; font-size: 25pt; color: Red; }
        #two { font-family: 'Monotype Corsiva'; font-size: 50pt; color: green; }
    </style>
</head>
<body>
    <p id = "one">Web Technology</p>
    <p id = "two">Web Technology</p>
</body>
```

4 what is array in javascript ? explain various ways of creating arrays , mention any 5 array method and explain their use?

An array in JavaScript is a data structure that allows you to store and organize multiple values in a single variable. These values can be of any data type, including numbers, strings, objects, functions, and even other arrays. JavaScript arrays are ordered collections, which means each element has an index, starting from 0 for the first element.

There are several ways to create arrays in JavaScript:

1. **Array Literal Notation:** The most common and straightforward way to create an array is by using square brackets `[]` and specifying its elements inside.

```
javascript
```

- `let fruits = ["apple", "banana", "cherry"];`

- **Array Constructor:** You can create an array using the `Array` constructor. While this is less common than using literal notation, it's another valid way.

```
javascript
• let colors = new Array("red", "green", "blue");
```

- **Array.from():** This method creates a new array from an iterable object or array-like structure, such as the arguments object or a NodeList.

```
javascript
• let arrayFromStr = Array.from("hello");
// Result: ["h", "e", "l", "l", "o"]
```

- **Array.of():** This method creates a new array with the provided elements, regardless of their type. It's useful for creating arrays with a single value.

```
javascript
• let singleValueArray = Array.of(5); // Creates an
array with one element [5]
```

- **Array with a Specific Length:** You can create an array with a specific length without initializing its elements. This is often used when you know how many elements you want but don't yet have the actual values.

```
javascript
  5. let emptyArray = new Array(3); // Creates an array
    with a length of 3, but no initial values
  6.
```

Now, let's discuss five common array methods and their uses:

1. **`push()` and `pop()`:**
    - `push()` adds one or more elements to the end of an array.
    - `pop()` removes and returns the last element from an array.

```
  javascript
• let numbers = [1, 2, 3];
numbers.push(4); // Adds 4 to the end of the array,
numbers is now [1, 2, 3, 4]
let lastNumber = numbers.pop(); // Removes and returns 4
```

- **`shift()` and `unshift()`:**

  - `shift()` removes and returns the first element from an array.
  - `unshift()` adds one or more elements to the beginning of an array.

```
javascript
• let colors = ["red", "green", "blue"];
let removedColor = colors.shift(); // Removes and
returns "red"
colors.unshift("orange"); // Adds "orange" to the
beginning, colors is now ["orange", "green", "blue"]
```

- **`concat()`:**

  - `concat()` combines two or more arrays, creating a new array without modifying the original arrays.

```
javascript
• let array1 = [1, 2];
let array2 = [3, 4];
let combinedArray = array1.concat(array2); // Creates
[1, 2, 3, 4]
```

- **`slice()`:**

  - `slice()` extracts a portion of an array into a new array. It takes two arguments: the start index and the end index (exclusive).

```
javascript
• let fruits = ["apple", "banana", "cherry", "date"];
let slicedFruits = fruits.slice(1, 3); // Creates a new
array ["banana", "cherry"]
```

- **`forEach()`:**

  - `forEach()` iterates through the elements of an array and applies a provided function to each element.

```
javascript
  5. let numbers = [1, 2, 3];
```

```
6. numbers.forEach(function (number) {
7.   console.log(number * 2); // Logs 2, 4, 6
8. });
9.
```

These are just a few of the many methods available for working with arrays in JavaScript. Arrays are a fundamental data structure in the language, and understanding how to create and manipulate them is essential for web development and many other applications.

3. Develop and demonstrate a XHTML file that includes Javascript script for the following problems:
a) Input : A number n obtained using prompt Output : The first n Fibonacci numbers
b) Input : A number n obtained using prompt Output : A table of numbers from 1 to n and their squares
using alert

Lab3a.html

```
<html lang="en">
        <head>
                <meta charset="UTF-8">
                <title>Lab3a</title>
        </head>
   <body>
      <script type="text/javascript">
         //initialize variables
         var fib1=0,fib2=1,fib=0;
         var n=prompt("Enter a number");
         if(n!=null && n>0)
            {
               document.write("<h1>First " + n + " Fibonacci numbers are:
</h1><br>");
               //if input is one number
               if(n==1)
                  document.write("<p>" + fib1 + "</p><br/>");
               //if input is two numbers
```

```
                document.write("<h1>First " + n + " Fibonacci numbers are:
</h1><br>");
                //if input is one number
                if(n==1)
                   document.write("<p>" + fib1 + "</p><br/>");
                //if input is two numbers
                else
                   document.write("<p>" + fib1 + "</p><br/><p>" + fib2 +
"</p><br/>");
                   //if input is more than two numbers, find the next Fibonacci
number
                   for(i=3;i<=n;i++)
                      {
                         fib=fib1+fib2;
                         document.write("<p>" + fib + "</p><br/>");
                         fib1=fib2;
                         fib2=fib;
                      }
             }
          else
             alert("Please enter proper input value");
       </script>
    </body>
 </html>
```

---

4 c discuss the following javascript method :

1 alert()

2 prompt()

3 confirm()

An array in JavaScript is a data structure that allows you to store and organize multiple values in a single variable. These values can be of any data type, including numbers, strings, objects, functions, and even other arrays. JavaScript arrays are ordered collections, which means each element has an index, starting from 0 for the first element.

There are several ways to create arrays in JavaScript:

1. **Array Literal Notation:** The most common and straightforward way to create an array is by using square brackets `[]` and specifying its elements inside.

```
javascript
• let fruits = ["apple", "banana", "cherry"];
```

• **Array Constructor:** You can create an array using the `Array` constructor. While this is less common than using literal notation, it's another valid way.

```
javascript
• let colors = new Array("red", "green", "blue");
```

• **Array.from():** This method creates a new array from an iterable object or array-like structure, such as the arguments object or a NodeList.

```
javascript
• let arrayFromStr = Array.from("hello");
// Result: ["h", "e", "l", "l", "o"]
```

• **Array.of():** This method creates a new array with the provided elements, regardless of their type. It's useful for creating arrays with a single value.

```
javascript
• let singleValueArray = Array.of(5); // Creates an
array with one element [5]
```

• **Array with a Specific Length:** You can create an array with a specific length without initializing its elements. This is often used when you know how many elements you want but don't yet have the actual values.

```
javascript
  5. let emptyArray = new Array(3); // Creates an array
     with a length of 3, but no initial values
  6.
```

Now, let's discuss five common array methods and their uses:

1. **`push()` and `pop()`:**
    - `push()` adds one or more elements to the end of an array.

- `pop()` removes and returns the last element from an array.

```
javascript
```

- ```
  let numbers = [1, 2, 3];
  numbers.push(4); // Adds 4 to the end of the array, numbers is now [1, 2, 3, 4]
  let lastNumber = numbers.pop(); // Removes and returns 4
  ```

- **`shift()` and `unshift()`:**

  - `shift()` removes and returns the first element from an array.
  - `unshift()` adds one or more elements to the beginning of an array.

```
javascript
```

- ```
  let colors = ["red", "green", "blue"];
  let removedColor = colors.shift(); // Removes and returns "red"
  colors.unshift("orange"); // Adds "orange" to the beginning, colors is now ["orange", "green", "blue"]
  ```

- **`concat()`:**

  - `concat()` combines two or more arrays, creating a new array without modifying the original arrays.

```
javascript
```

- ```
  let array1 = [1, 2];
  let array2 = [3, 4];
  let combinedArray = array1.concat(array2); // Creates [1, 2, 3, 4]
  ```

- **`slice()`:**

  - `slice()` extracts a portion of an array into a new array. It takes two arguments: the start index and the end index (exclusive).

```
javascript
```

- ```
  let fruits = ["apple", "banana", "cherry", "date"];
  let slicedFruits = fruits.slice(1, 3); // Creates a new array ["banana", "cherry"]
  ```

- **`forEach()`:**
  - `forEach()` iterates through the elements of an array and applies a provided function to each element.

```
javascript
  5. let numbers = [1, 2, 3];
  6. numbers.forEach(function (number) {
  7.   console.log(number * 2); // Logs 2, 4, 6
  8. });
  9.
```

These are just a few of the many methods available for working with arrays in JavaScript. Arrays are a fundamental data structure in the language, and understanding how to create and manipulate them is essential for web development and many other applications.

5-A) What Is Bootstrap? Features, File Structure Of Bootstrap. How To Include Bootstrap In Xhtml Structure?

I. *Bootstrap introduction*

A. Bootstrap is a popular open-source front-end framework that is used to create responsive and mobile-first websites and web applications.

B. ***Mark Otto*** **and** ***Jacob Thornton*** developed the Bootstrap, at *Twitter*. In *August 2011*, Bootstrap was released as an open source product, on GitHub.

C. It provides a collection of CSS and JavaScript components, such as grids, forms, buttons, navigation bars, and more, which can be easily implemented and customized to create responsive and visually appealing web interfaces.

D. With Bootstrap, developers can save time and effort by utilizing pre-designed components, as well as the grid system for creating responsive layouts.

E. It also provides numerous styling options and utilities to enhance the overall appearance and functionality of websites.

F. Bootstrap is widely used by web developers to streamline the web development process and create consistent and visually appealing user interfaces.

II. *Bootstrap features*

There are several benefits of using Bootstrap:

1. **Responsive Design:** Bootstrap is mobile-first, adapting to various screen sizes for a consistent user experience on different devices.
2. **Time-Saving:** It offers ready-made CSS and JavaScript components, saving developers time by avoiding building everything from scratch.
3. **Consistent Appearance:** Achieve a professional design with predefined styles and themes that can be customized to match your brand.
4. **Cross-Browser Compatibility:** Bootstrap functions smoothly across different browsers, ensuring a consistent experience for all users.
5. **Community and Support:** Benefit from a large community of developers who offer support and resources through forums and online platforms.

6. **Accessibility:** Bootstrap adheres to accessibility guidelines, making your application usable for people with disabilities.
7. **Continuous Updates:** Regularly updated with new features, bug fixes, and optimizations to keep your application current and efficient.

B. Discuss the following using a Bootstrap code snippet:

a) Table In Bootstrap

In Bootstrap, creating tables is a straightforward process. Bootstrap provides classes and styles that allow you to design and customize tables easily. Here are some notes on using tables in Bootstrap:

I. Basic Table Structure:

To create a basic table in Bootstrap, you can use the following structure:

```
html
Copy code
<table class="table">
    <thead>
        <tr>
            <th>Header 1</th>
            <th>Header 2</th>
            <th>Header 3</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Row 1, Column 1</td>
            <td>Row 1, Column 2</td>
            <td>Row 1, Column 3</td>
        </tr>
        <!-- Additional rows here -->
    </tbody>
</table>
```

II. Striped Tables:

To create striped tables for better readability, add the table-striped class to the table element:

```
html
Copy code
<table class="table table-striped">
```

```
    <!-- Table content -->
</table>
```

III. Bordered Tables:

To add borders to the table and its cells, use the table-bordered class:

html
Copy code

```
<table class="table table-bordered">
    <!-- Table content -->
</table>
```

IV. Hover Effect:

To highlight rows when hovering over them, apply the table-hover class:

```
html
Copy code
<table class="table table-hover">
    <!-- Table content -->
</table>
```

V. Responsive Tables:

For tables that need to be horizontally scrollable on smaller screens, use the table-responsive class:

```
html
Copy code
<div class="table-responsive">
    <table class="table">
        <!-- Table content -->
    </table>
</div>
```

VI. Contextual Classes:

You can apply contextual classes to table rows or cells to indicate different states, such as success, info, warning, or danger. Here's an example:

```
html
Copy code
<table class="table">
    <tbody>
        <tr class="table-success">
            <td>Success</td>
        </tr>
        <tr class="table-info">
            <td>Info</td>
        </tr>
        <tr class="table-warning">
            <td>Warning</td>
```

```
        </tr>
        <tr class="table-danger">
            <td>Danger</td>
        </tr>
    </tbody>
</table>
```

These are the basic techniques for creating and customizing tables in Bootstrap. You can further style and format tables as per your project's requirements by combining classes and CSS rules.

b) images

```html
<html>
<head>
    <title>Image Styling Examples</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css
        /bootstrap.min.css">
</head>
<body>

<div class="container">
    <h2>Image Styling Examples</h2>
    <!-- Rounded Image -->
    <h3>Rounded Image</h3>
    <img src="image.jpg" alt="Rounded Image" class="img-rounded">

    <!-- Circular Image -->
    <h3>Circular Image</h3>
    <img src="image.jpg" alt="Circular Image" class="img-circle">

    <!-- Thumbnail Image -->
    <h3>Thumbnail Image</h3>
    <img src="image.jpg" alt="Thumbnail Image" class="img-thumbnail">

    <!-- Responsive Image -->
    <h3>Responsive Image</h3>
    <img src="image.jpg" alt="Responsive Image" class="img-responsive">
</div>
</body>
</html>
```

- Bootstrap has an **img class** that allows users to set various properties for images and display them responsively.
- The class enables images to be scaled according to the screen size while making sure the images do not exceed the size ofthe parent class.
- The image class has various available properties, some of which are demonstrated below:

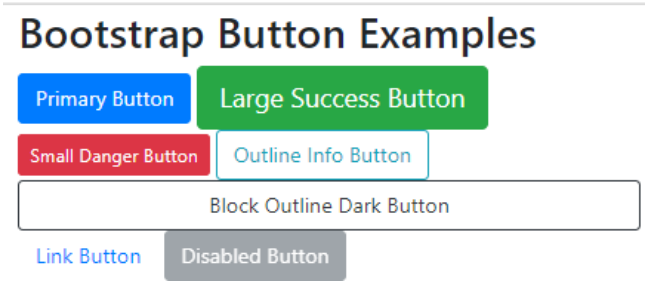
Class	Description
.img-rounded	Adds rounded corners to an image (not available in IE8)
.img-circle	Shapes the image to a circle (not available in IE8)
.img-thumbnail	Shapes the image to a thumbnail
.img-responsive	Makes an image responsive (will scale nicely to the parent element)

c) button

1. .btn: This is the base class for creating buttons in Bootstrap. You can add this class to any <button> or <a> element to create a button.
2. Button Styles:
    - .btn-primary: Creates a button with a primary blue color.
    - .btn-secondary: Creates a button with a secondary gray color.
    - .btn-success: Creates a button with a success green color.
    - .btn-danger: Creates a button with a danger red color.
    - .btn-warning: Creates a button with a warning yellow color.
    - .btn-info: Creates a button with an info teal color.
    - .btn-light: Creates a button with a light gray color.
    - .btn-dark: Creates a button with a dark gray color.
    - .btn-link: Creates a button that looks like a link.
3. Button Sizes:
    - .btn-sm: Creates a small-sized button.
    - .btn-lg: Creates a large-sized button.
4. Block Level Buttons:
    - .btn-block: Makes the button span the full width of its parent container.
5. Outline Buttons:
    - .btn-outline-primary: Creates an outlined primary button.
    - .btn-outline-secondary: Creates an outlined secondary button.
    - You can use the corresponding outline classes with other styles as well (e.g., .btn-outline-success, .btn-outline- danger).
6. Active and Disabled Buttons:
    - .active: Adds the "active" state to a button.
    - .disabled: Disables a button, preventing interaction.

```html
<!DOCTYPE html>
<html>
<head>
    <title>Bootstrap Button Examples</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>

<div class="container">
    <h2>Bootstrap Button Examples</h2>

    <button class="btn btn-primary">Primary Button</button>
    <button class="btn btn-success btn-lg">Large Success Button</button>
    <button class="btn btn-danger btn-sm">Small Danger Button</button>

    <button class="btn btn-outline-info">Outline Info Button</button>
    <button class="btn btn-outline-dark btn-block">Block Outline Dark Button</button>

    <button class="btn btn-link">Link Button</button>
    <button class="btn btn-secondary disabled">Disabled Button</button>
</div>

</body>
</html>
```

d) Bootstrap Progress Bar:



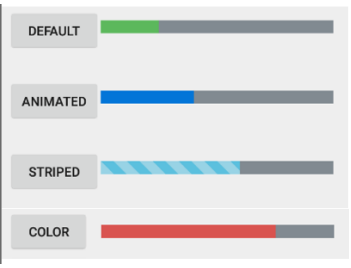
A Bootstrap progress bar is a visual indicator that shows the progress of a task or operation. It's commonly used to give users feedback on how much of a process has been completed.

VII. Key Points:

1. **Basic Progress Bar:** Bootstrap provides a simple, colored progress bar that can be customized according to your design.
2. **Striped Progress Bar:** You can add a striped pattern to the progress bar using the **.progress-bar-striped** class.
3. **Animated Progress Bar:** To make the striped pattern animate, use the **.progress-bar-animated** class in addition to **.progress-bar-striped**.
4. **Width:** The width of the progress bar is determined by the **style** attribute's **width** property.
5. **Color Classes:** Bootstrap offers color classes like **.bg-success**, **.bg-info**, **.bg-warning**, and **.bg-danger** to change the color of the progress bar.

Example:

```
<body>
<div class="container">
    <h2>Bootstrap Progress Bar Example</h2>

    <!-- Basic Progress Bar -->
    <div class="progress">
        <div class="progress-bar" style="width: 50%;">50%</div>
    </div>

    <!-- Striped Progress Bar -->
    <div class="progress">
        <div class="progress-bar progress-bar-striped" style="width: 70%;">70%</div>
    </div>

    <!-- Animated Progress Bar -->
    <div class="progress">
        <div class="progress-bar progress-bar-striped progress-bar-animated" style="width: 80%;">80%</div>
    </div>

    <!-- Colored Progress Bar -->
    <div class="progress">
        <div class="progress-bar bg-success" style="width: 90%;">90%</div>
    </div>
</div>
</body>
```



6-A) List out the various types of forms in Bootstrap and provide a code snippet.

1. ***Bootstrap Forms***

Bootstrap Form Layouts

Bootstrap provides three types of form layouts:

- Vertical form (this is default)
- Horizontal form
- Inline form

Standard rules for all three form layouts:

- Wrap labels and form controls in (needed for optimum spacing)
- Add class .form-control to all textual <input>, <textarea>, and <select> elements

I. Bootstrap Vertical Form (default)

Email:

Password:

Remember me

Submit

The following example creates a vertical form with two input fields, one checkbox, and a submit button:

Example

```
<form action="/action_page.php">
  <div class="form-group">
    <label for="email">Email address:</label>
    <input type="email" class="form-control" id="email">
  </div>
  <div class="form-group">
    <label for="pwd">Password:</label>
    <input type="password" class="form-control" id="pwd">
  </div>
  <div class="checkbox">
    <label><input type="checkbox"> Remember me</label>
```

```
  </div>
  <button type="submit" class="btn btn-default">Submit</button>

</form>
```

II. Bootstrap Inline Form

Email:

Password:

Remember me

Submit

In an inline form, all of the elements are inline, left-aligned, and the labels are alongside.

Note: This only applies to forms within viewports that are at least 768px wide!

Additional rule for an inline form:

- Add class .form-inline to the <form> element

The following example creates an inline form with two input fields, one checkbox, and one submit button:

Example

```
<form class="form-inline" action="/action_page.php">
  <div class="form-group">
    <label for="email">Email address:</label>
    <input type="email" class="form-control" id="email">
  </div>
  <div class="form-group">
    <label for="pwd">Password:</label>
    <input type="password" class="form-control" id="pwd">
  </div>
  <div class="checkbox">
    <label><input type="checkbox"> Remember me</label>
  </div>
  <button type="submit" class="btn btn-default">Submit</button>
</form>
```

III. Bootstrap Horizontal Form

Email:

Password:

Remember me

Submit

A horizontal form means that the labels are aligned next to the input field (horizontal) on large and medium screens. On small screens (767px and below), it will transform to a vertical form (labels are placed on top of each input).

Additional rules for a horizontal form:

- Add class .form-horizontal to the <form> element
- Add class .control-label to all <label> elements

Tip: Use Bootstrap's predefined grid classes to align labels and groups of form controls in a horizontal layout.

The following example creates a horizontal form with two input fields, one checkbox, and one submit button.

Example

```
<form class="form-horizontal" action="/action_page.php">
 <div class="form-group">
  <label class="control-label col-sm-2" for="email">Email:</label>
  <div class="col-sm-10">
   <input type="email" class="form-control" id="email" placeholder="Enter
email">
  </div>
 </div>
 <div class="form-group">
  <label class="control-label col-sm-2" for="pwd">Password:</label>
  <div class="col-sm-10">
   <input type="password" class="form-control" id="pwd" placeholder="Enter
password">
  </div>
 </div>
 <div class="form-group">
  <div class="col-sm-offset-2 col-sm-10">
```

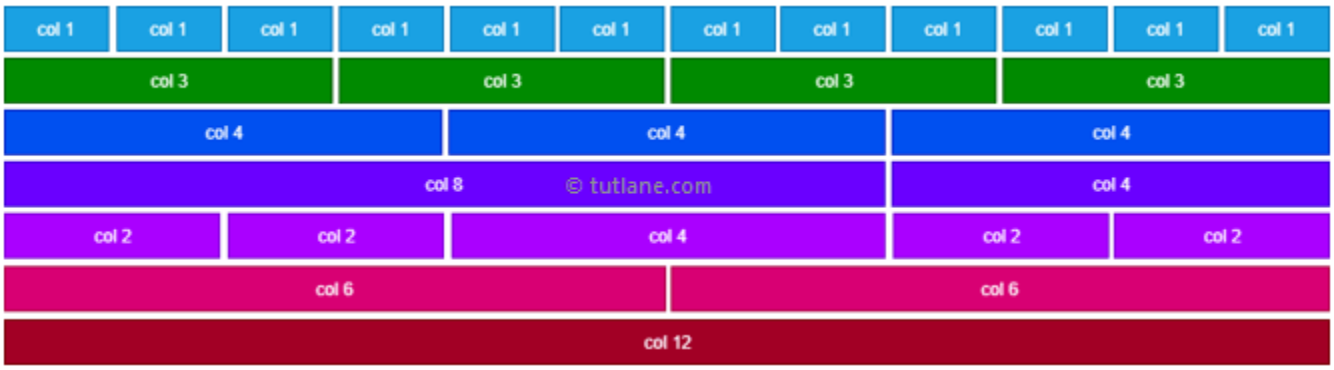
```
    <div class="checkbox">
      <label><input type="checkbox"> Remember me</label>
    </div>
   </div>
  </div>
  <div class="form-group">
   <div class="col-sm-offset-2 col-sm-10">
    <button type="submit" class="btn btn-default">Submit</button>
   </div>
  </div>
</form>
```

B-) Discuss briefly the Grid system of Bootstrap.

In bootstrap, the **grid system** is useful for building quick web page layouts that are responsive to the different devices based on screen sizes. In bootstrap 4, the grid system is built with a mobile-first flexbox, and it allows up to **12** columns across the page.

The bootstrap grid system will use a series of containers, rows, and columns to define the layout and to align the content appropriately based on the device. In the grid system, you can add up to 12 columns and add as many rows as you like, and the columns will re-arrange automatically based on the device screen size.

Using a bootstrap grid system, we can create a responsive web page layout by defining the **12** columns individually or by grouping the columns to create wider columns.



2. Bootstrap Grid Classes

Bootstrap 4 has included 5 predefined grid classes to scale the content depending on the device or viewport size.

- .col-*
- .col-sm-*
- .col-md-*

- .col-lg-*
- .col-xl-*

Here, the asterisk (*) is the span width of the column from **1** to **12**. The following table lists how the grid system classes will work across multiple devices.

Class	Device Type	Width
.col-*	Extra Small	<576px
.col-sm-*	Small	≥576px
.col-md-*	Medium	≥768px
.col-lg-*	Large	≥992px
.col-xl-*	Extra Large	≥1200px

3. Structure of Bootstrap Grid

To create responsive web page layouts using a bootstrap grid system, we need to use rows and columns within the container or container-fluid like as shown below.

```
<div class="container">
   <div class="row">
     <div class="col-*-*"></div>
     <div class="col-*-*"></div>
     <div class="col-*-*"></div>
   </div>
</div>
```

C-) Explain Bootstrap progress bars with a code snippet. Explain their use.

A progress bar can be used to show how far a user is in a process.

To create a default progress bar, add a `.progress` class to a container element and add the `.progress-bar` class to its child element. Use the CSS `width` property to set the width of the progress bar:

Example

```
<div class="progress">
  <div class="progress-bar" style="width:70%"></div>
</div>
```

I. Progress Bar Height

The height of the progress bar is `1rem` (usually `16px`) by default. Use the CSS `height` property to change it:

Example

```
<div class="progress" style="height:20px">
  <div class="progress-bar" style="width:40%;"></div>
</div>
```

II. Progress Bar Labels

Add text inside the progress bar to show the visible percentage:

Example

```
<div class="progress">
  <div class="progress-bar" style="width:70%">70%</div>
</div>
```

III. Colored Progress Bars

By default, the progress bar is blue (primary). Use any of the contextual background classes to change its color:

Example

```
<!-- Blue -->
<div class="progress">
  <div class="progress-bar" style="width:10%"></div>
</div>

<!-- Green -->
<div class="progress">
  <div class="progress-bar bg-
success" style="width:20%"></div>
</div>

<!-- Turquoise -->
<div class="progress">
  <div class="progress-bar bg-info" style="width:30%"></div>
</div>

<!-- Orange -->
<div class="progress">
   <div class="progress-bar bg-
warning" style="width:40%"></div>
</div>

<!-- Red -->
<div class="progress">
  <div class="progress-bar bg-
danger" style="width:50%"></div>
</div>

<!-- White -->
<div class="progress border">
  <div class="progress-bar bg-white" style="width:60%"></div>
</div>

<!-- Grey -->
<div class="progress">
  <div class="progress-bar bg-
secondary" style="width:70%"></div>
```

```
</div>

<!-- Light Grey -->
<div class="progress border">
  <div class="progress-bar bg-light" style="width:80%"></div>
</div>

<!-- Dark Grey -->
<div class="progress">
  <div class="progress-bar bg-dark" style="width:90%"></div>
</div>
```

IV. Striped Progress Bars

Use the `.progress-bar-striped` class to add stripes to the progress bars:

Example

```
<div class="progress">
  <div class="progress-bar progress-bar-
striped" style="width:40%"></div>
</div>
```

V. Animated Progress Bar

Add the `.progress-bar-animated` class to animate the progress bar:

VI. Example

```
<div class="progress-bar progress-bar-striped progress-bar-
animated" style="width:40%"></div>
```

VII. Multiple Progress Bars

Progress bars can also be stacked:

Example

```
<div class="progress">
  <div class="progress-bar bg-success" style="width:40%">
```

```
    Free Space
  </div>
  <div class="progress-bar bg-warning" style="width:10%">
    Warning
  </div>
  <div class="progress-bar bg-danger" style="width:20%">
    Danger
  </div>
</div>
```

Q7 a. What is Querry? What are the advantages of jQuerry? Explain the syntax of Querry script with a suitable example.

Answer:-

• jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as

well as event handling, CSS animation, and Ajax.

• jQuery is a lightweight, "write less, do more", JavaScript library.

• Using raw JavaScript can result in dozens of lines of code.

• The creators of jQuery specifically created the library to make common tasks trivial.

• The real power in this jQuery statement comes from the selector, an expression for identifying

target elements on a page that allows us to easily identify and grab the elements we need.

Advantages:-

 Using raw JavaScript can result in dozens of lines of code for each of these tasks.

 The creators of jQuery specifically created the library to make common tas ks trivial. For

example, designers will use JavaScript to “zebra-stripe” tables— highlighting every other row in

a table with a contrasting color—taking up to 10 lines of code or more. Here’s how we

accomplish it using jQuery:

$("table tr:nth-child(even)").addClass("striped");

 jQuery statements to make your pages come alive.

 The real power in this jQuery statement comes from the selector, an expression for identifying

target elements on a page that allows us to easily identify and grab the elements we need

Example

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


<script type="text/javascript">
window.onload = function() {
document.getElementById('testButton').onclick = makeItRed;
};
function makeItRed() {
document.getElementById('xyz').style.color = 'red';
}
</script>
```

Q7 b. Develop JQuery programs to implement the following jQuerry effects:

(1) Show and hide ()

(2) fadein() and fadeout()

Answer:-

(1) Show and hide ()

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

```
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scri
pt>
  <script>
    $(document).ready(function(){
      $("#hide").click(function(){
        $("p").hide();
      });
      $("#show").click(function(){
        $("p").show();
      });
    });
  </script>
</head>
<body>

<p>If you click on the "Hide" button, I will disappear.</p>

<button id="hide">Hide</button>
<button id="show">Show</button>

</body>
</html>
```

(2) fadein() and fadeout()

```
<!DOCTYPE html>
<html>
<head>
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script>
```

```
    $(document).ready(function(){
      $("#fadeout").click(function(){
        $("#div1").fadeOut();
      });
      $("#fadein").click(function(){
        $("#div1").fadeIn();
      });
    });
  </script>
  <style>
    #div1 {
      width: 150px;
      height: 150px;
      display: none;
      background-color: red;
    }
  </style>
</head>
<body>

<button id="fadeout">Fade out</button>
<button id="fadein">Fade in</button>

<div id="div1"></div>
```

</body>

</html>

Q8. a. What is JQuery HTML? What are the methods used for DOM manipulation? Develop a Query program to get attribute values.

Answer:-

jQuery is an open source JavaScript library that simplifies the interactions between an HTML/CSS document, or more precisely the Document Object Model (DOM), and JavaScript.
Elaborating the terms, jQuery simplifies HTML document traversing and manipulation, browser event handling, DOM animations, Ajax interactions, and cross-browser JavaScript development.

We can create DOM elements on the fly by passing the $() function a string that contains the HTML

markup for those elements. For example, we can create a new paragraph element as follows:

$(" Hi there! ")

But creating a disembodied DOM element (or hierarchy of elements) isn't all that useful; usually the

element hierarchy created by such a call is then operated on using one of jQuery's DOM manipulation

functions.

<html>

<head>

<title>Follow me ! </title>

‹script type="text/javascript" src="

../scripts/jquery-1.2 js" >

</script>

<script type="text/javascript"›

Ready handler that creates HTML element

$ (function () {

sA<pvHE there ! 1） .insextatter （"#Eo12omlE"） ;

1) ;

</script>

</head>

‹body>

</body>

</html>

Existing element to be followed

Follow me!

This example establishes an existing HTML paragraph element named followMe in the document body.

In the script element within the <head> section, we establish a ready handler that uses the following

statement to insert a newly created paragraph into the DOM tree after the existing element:

$(" Hi there! ").insertAfter("#followMe");

b. What is an event? List the common events found in jQuery. Devalop a JQuery program to implement mouse enter( ) JQuery event.

Answer:-

Events are associated with different HTML elements. e.g. the click event is associated with button element; similarly keypress event is associated with text box or text area element. AngularJS provides multiple events which are associated with HTML control.

Click event

ng-click = "expression"

ng-click = "expression" defines a click event. When a button is clicked, an event occurs, and evaluates the expression. The click event normally works on button.

Double Click event

ng-dblclick = "expression"

ng-click = "expression" defines a double click event. When a button is double clicked, an event occurs, and evaluates the expression.

Double click event normally works on button.

Mouse move event

ng-mousemove = "expression"

ng-mousemove = "expression" defines a mouse move event. When the mouse moves, an event occurs, and evaluates the expression.

Mouse move event normally works on div, body and specific area or element.

"ng-mousemove" defines an AngularS mouse move event.

"ng-mouseover" defines an AngularJS mouse over event.

"ng-mouseleave" defines an Angular)S mouse leave event.

"ng-keyup" defines an AngularIS key up event.

"ng-keydown" defines an AngularJS key down event.

<!DOCTYPE html>

<html>

```
<head>
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script>
    $(document).ready(function(){
      $("#target").mouseenter(function(){
        $(this).css("background-color", "lightgray");
      });
    });
  </script>
  <style>
    #target {
      width: 200px;
      padding: 20px;
      text-align: center;
      background-color: lightblue;
      cursor: pointer;
    }
  </style>
</head>
<body>

<div id="target">
  Mouse over me
</div>
```

</body>

</html>

Q.9

a. What is Angular JS? Explain the following Angular JS directives:

(i) ng_app (ii) ng_model (iii) ng_bind

The AngularJS is a framework of JavaScript. It can use HTML as a template language and can extend HTML's sentence structure to state an application's components plainly and briefly.

The syntax of AngularJS looks like this:

```
<div ng-app=" "> .....
</div>
```

We know that **div** is an html tag, but **ng-app** is directive of Angular JS, which is used in div tag like an attribute. (We will discuss about directive latter)

AngularJS Directives

AngularJS lets you extend HTML with new attributes called **Directives**.
AngularJS has a set of built-in directives which offers functionality to your applications.
AngularJS also lets you define your own directives.
They are App Directive, Model Directive, Bind Directive, Init Directive, and Repeat Directive.

App Directive

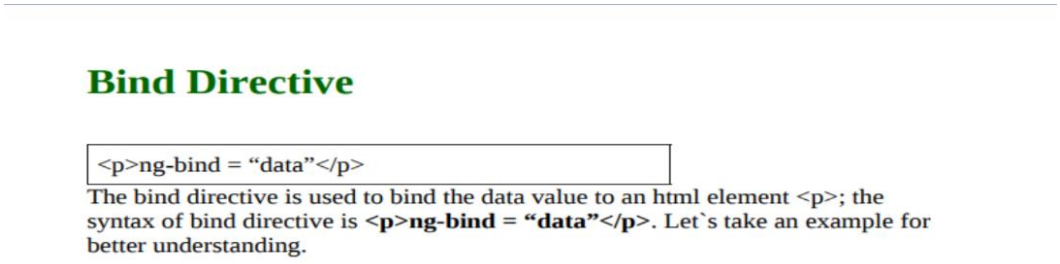
```
ng-app= " "
```

The app directive defines the area of AngularJS application. The syntax of app directive is **ng-app = " "**; In here the **ng** is the namespace of AngularJS and **app** is the application area of Angular JS.

Model Directive

```
ng-model = "data"
```

The model directive is used to bind the inputted value from HTML controls (input, checkbox and select etc.) to application data. The **ng-model = "data"** is the syntax of model directive. Let`s take an example for better understanding.



b. Write an Angular JS program to use expressions.

```
<!DOCTYPE html>
<html ng-app="myApp">

<head>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>

<body>

  <div ng-controller="myCtrl">
    <p>Enter a number: <input type="number" ng-model="number"></p>
    <p>Result: {{ number * 2 }}</p>

    <p>{{ greeting }}</p>

    <p>{{ person.name }} is {{ person.age }} years old.</p>
```

```
    <ul>
      <li ng-repeat="item in items">{{ item }}</li>
    </ul>

    <p>{{ stringExpression }}</p>
  </div>

  <script>
    var app = angular.module('myApp', []);
    app.controller('myCtrl', function($scope) {
      $scope.number = 0;
      $scope.greeting = "Hello, AngularJS!";
      $scope.person = {
        name: "John",
        age: 30
      };
      $scope.items = ["Apple", "Banana", "Cherry", "Date"];
      $scope.stringExpression = "This is a string expression.";
    });
  </script>

</body>

</html>
```

c. Briefly discuss the use of filter in Angular JS with an example.

What is filter?

Filter is used to format the value of data. The pipe sign (|) indicates that filter is used. The proper syntax of filter looks like this:

```
Value | filter
```

1. Lowercase
2. Uppercase
3. Number
4. Currency
5. JSON

Uppercase filter

```
Value | uppercase
```

The uppercase filter changes the text to upper case. Suppose a user writes a text in lower case (e.g. ray) or title case (e.g. Ray) or in mixed case (e.g. rAy or RaY or rAY etc.), and you want the upper case result, then you will have to use upper case filter.

Lowercase filter

```
Value | lowercase
```

The lowercase filter changes the text to lower case. Suppose a user writes a text in upper case (e.g. RAY YAO) or title case (e.g. Ray Yao) or in mixed case (e.g. rAy or RaY or rAY etc.), and you want the lower case result, then you will have to use lower case filter.

AngularJS Number Filter

In angularjs, **number** filter is used to format the number and convert it as string or text. By using number filter in angularjs we can show number with decimal values and we define limit to show number of decimal values.

Currency filter

```
Value | currency
```

The currency filter is used to display the result in currency format.

JSON Filter

Display a JavaScript object as a JSON string:

```
<!DOCTYPE html>

<html ng-app="myApp">


<head>
```

```
 <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script
>
</head>

<body>

 <div ng-controller="myCtrl">
  <p>Original text: {{ originalText }}</p>
  <p>Uppercase: {{ originalText | uppercase }}</p>
  <p>Lowercase: {{ originalText | lowercase }}</p>
  <p>Number: {{ number | number:2 }}</p>
  <p>JSON: {{ object | json }}</p>
  <p>Formatted Currency: {{ currencyValue | currency }}</p>
 </div>

 <script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function($scope) {
   $scope.originalText = 'Hello AngularJS';
   $scope.number = 123.45678;
   $scope.object = { name: 'John', age: 30 };
   $scope.currencyValue = 123.45;
  });
 </script>
```

```
</body>

</html>
```

Q.10

a. What is a Angular JS Service? Explain any three of them by using code snippet.

In Angular JS, a service is a singleton object that performs a specific task and can be used throughout an application. It is used to organize and share code across your app.

Certainly, here's an example that demonstrates the use of the $http, $location, $timeout service in a single AngularJS program:

```
<!DOCTYPE html>

<html ng-app="myApp">


<head>

  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script
>

</head>


<body>


  <div ng-controller="myCtrl">
```

```
  <p>Current URL: {{ currentUrl }}</p>
  <ul>
    <li ng-repeat="user in users">
      {{ user.name }}
    </li>
  </ul>
  <p ng-hide="hiddenMessage">{{ message }}</p>
</div>

<script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function($scope, $http, $location, $timeout) {
    $http.get('https://jsonplaceholder.typicode.com/users')
      .then(function(response) {
        $scope.users = response.data;
      });

    $scope.currentUrl = $location.absUrl();

    $scope.message = 'This message will disappear in 3 seconds!';
    $timeout(function() {
      $scope.hiddenMessage = true;
    }, 3000);
  });
</script>
```

```
</body>


</html>
```

b. Write an AngularJS program to demonstrate client-side form validation.

```
<!DOCTYPE html>
<html ng-app="myApp">


<head>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script
>
</head>


<body>


  <div ng-controller="myCtrl">
    <form name="myForm" novalidate>
      <p>Username:</p>
      <input type="text" name="username" ng-model="user.username" required>
      <span style="color:red" ng-show="myForm.username.$touched &&
myForm.username.$invalid">
        <span ng-show="myForm.username.$error.required">Username is
required.</span>
      </span>
      <br><br>
```

```
    <p>Email:</p>
    <input type="email" name="email" ng-model="user.email" required>
    <span style="color:red" ng-show="myForm.email.$touched &&
myForm.email.$invalid">
      <span ng-show="myForm.email.$error.required">Email is required.</span>
      <span ng-show="myForm.email.$error.email">Invalid email address.</span>
    </span>
    <br><br>
    <button ng-disabled="myForm.$invalid" ng-
click="submitForm()">Submit</button>
  </form>
 </div>

 <script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function($scope) {
    $scope.user = {};

    $scope.submitForm = function() {
      alert('Form submitted successfully.');
    };
  });
 </script>

</body>
```

</html>