Web Technology
Assessment Solution
2023 Fall
NCIT

Sanjaya (Bir Bikram) Shrestha

February 27, 2023

Contents

1 (Q No.1 A) To make an online presence or just transfer any data there are certain predefined rules to follow. Explain any 5 such commonly used protocols that serve different purposes of their own.

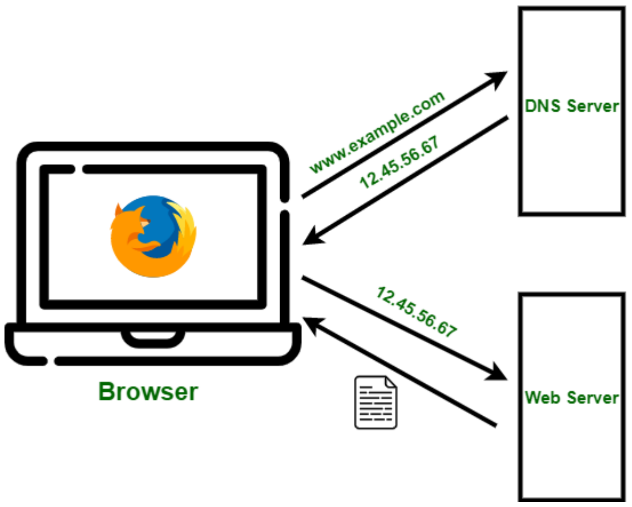
1. **TCP/IP: (Transmission Control Protocol/Internet Protocol)** This is the foundation protocol of the internet and is used for communication between devices over the internet. The web browser sends a request to the website's server using the TCP/IP protocol and then server sends the requested web page back to the web browser using TCP/IP as well.
2. **HTTP: (Hyper Text Transfer Protocol)** This is the protocol used for web browsing and transferring web pages from servers to clients. The browser sends http request to the server hosting the website. The server responds with an HTTP response, which includes the requested web page.
3. **FTP: File transfer protocol** This is the protocol used for file transfer over the internet. When we need to transfer a large file from one computer to another over the internet, you can use the FTP protocol. We need an FTP client program to connect to the FTP server where the file is stored, and then we can download the file using the FTP protocol.
4. **SMTP: (Simple Mail Transfer Protocol)** This is the protocol used for email transmission between servers and clients. We are sending an email from your email client program to someone else's email address. Our email client uses the SMTP protocol to send the email to the email provider's SMTP server. The SMTP server then relays the email to the recipient's email provider using SMTP as well.
5. **DNS: (Domain Name System)** This is the protocol used for translating domain names into IP addresses. When we type in a website URL into your web browser's address bar, the browser sends a request to a DNS server to look up the IP address associated with that URL. The DNS server responds with the IP address, which the web browser then uses to connect to the website's server.
6. **SSH: (Secure Shell)** This is the protocol used for secure remote login and file transfer. If you need to remotely access a computer or server securely, you can use the SSH protocol. You'll need an SSH client program to connect to the remote computer or server, and then you can securely log in and transfer files using SSH.
7. **DHCP: (Dynamic Host Control Protocol)** This is the protocol used for automatically assigning IP addresses to devices on a network. When a device joins a network, it can use DHCP to automatically obtain an IP address from the network's DHCP server. This allows devices to join a network without needing to manually configure IP addresses.
8. **POP: (Post Office Protocol)** This is the protocol used for retrieving email from a server. We using an email client program like Microsoft Outlook or Apple Mail to download our email, we're likely using the POP protocol. POP allows email client to download all of your email from email provider's server and store it on local computer.

2 (Q No.1B) What happens when you enter https://ncit.edu.np into your browser? Describe along with its block diagram.

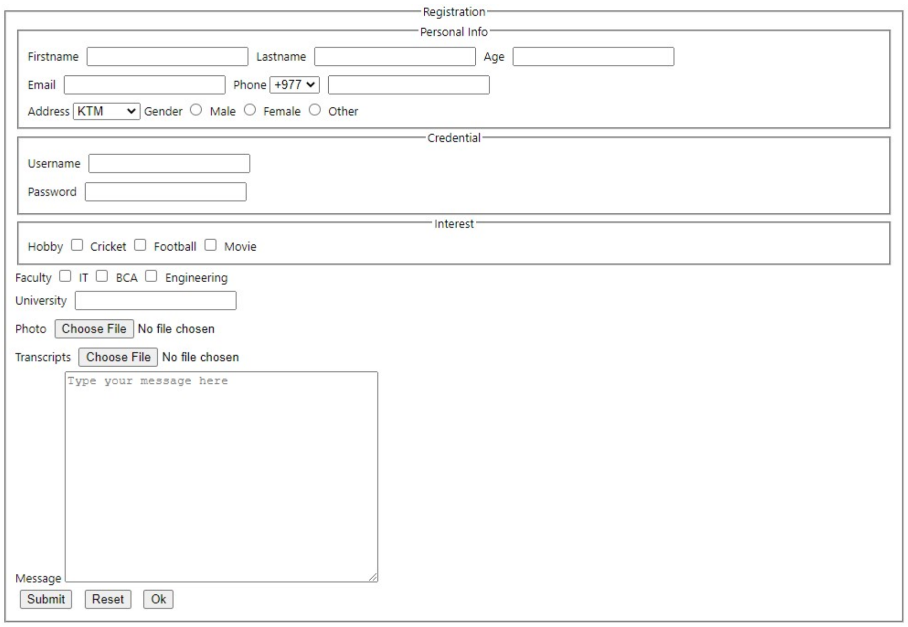
When we type https://ncit.edu.np into our browser following steps will happen:

1. Browser will initiates a request to the Domain Name System (DNS) to resolve the domain name "https://ncit.edu.np" into an IP address.
2. The DNS responds with the IP address associated with the domain name, which in this case is the IP address of the server hosting the website.
3. Browser will initiates a secure connection with the server using HTTPS protocol, which includes a series of handshaking and authentication steps to ensure the connection is secure and encrypted.
4. The server responds with the HTML code for the home page of the pu.edu.np website.
5. Browser then receives the HTML code and begins to parse it to render the page.
6. The browser sends additional requests for any external files referenced in the HTML code, such as CSS stylesheets, images, and scripts.
7. The server responds with the requested files and the browser uses them to render the page as intended.
8. Then if we interact with the website, such as clicking a link or submitting a form, the browser sends additional requests to the server and the server responds with the appropriate content.

```
                            +---------------------+
                            | DNS Resolution      |
                            +---------------------+
                                        |
                                        v
    +------------+          +-----------------------+
    | Browser    |          | HTTPS Handshaking      |
    +------------+          +-----------------------+
            |                         |
            v                         v
    +-------------------+  +---------------------------+
    | Receive HTML code |  | Receive external resources |
    +-------------------+  +---------------------------+
            |                         |
            v                         v
    +-------------------+  +------------------------+
    | Parse HTML code   |  | Render page with assets |
    +-------------------+  +------------------------+
            |                         |
            v                         v
    +------------------+  +------------------+
    | Interact with website           |
    +------------------+  +------------------+
```



3 (Q No.2A) Write HTML code for the following



Registration

Personal Info

Firstname Lastname Age

Email Phone +977

Address KTM Gender Male Female Other

Credential

Username

Password

Interest

Hobby Cricket Football Movie

Faculty IT BCA Engineering

University

Photo Choose File No file chosen

Transcripts Choose File No file chosen

Type your message here

Message

Submit Reset Ok

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        input {
            margin: 5px;
        }
        fieldset {
            border: 1px dashed black;
        }

    </style>
</head>
<body>
    <form action="">
        <fieldset>
            <legend align="center">Registration</legend>
            <fieldset>
                <legend align="center">Personal Info</legend>
                <label for="firstname">Firstname</label>
                <input type="text" name="firstname" id="firstname">

                <label for="lastname">Lastname</label>
                <input type="text" name="lastname" id="lastname">

                <label for="age">Age</label>
                <input type="number" name="age" id="age"> <br>

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

                <label for="phone_number">Phone </label>
                <select name="phone_code" id="phone_code">
                    <option value="nep" selected>+977</option>
                    <option value="india">+91</option>
                    <option value="us">+1</option>
                </select>
                <input type="phone_number" name="phone_number" id="phone_number">
    <br>

                <label for="address">Address </label>
                <select name="address" id="address">
                    <option value="ktm" selected>KTM</option>
                    <option value="pokhara">Pokhara</option>
                    <option value="butwal">Butwal</option>
                </select>

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


            </fieldset>

            <fieldset>
                <legend align="center">Credential</legend>

                <label for="username">Username</label>
                <input type="text" name="username" id="username"> <br>

                <label for="password">Password</label>
                <input type="text" name="password" id="password">

            </fieldset>

            <fieldset>
                <legend align="center">Interest</legend>

                <label for="interest">Hobby</label>
                <input type="checkbox" name="interest" id="interest" value="
    cricket"> Cricket
                <input type="checkbox" name="interest" id="interest" value="
    football"> Football
                <input type="checkbox" name="interest" id="interest" value="movie"
    > Movie

            </fieldset>

            <label for="faculty">Faculty</label>
            <input type="checkbox" name="faculty" id="faculty" value="it"> IT
            <input type="checkbox" name="faculty" id="faculty" value="bca"> BCA
            <input type="checkbox" name="faculty" id="faculty" value="engineering"
    > Engineering <br>

            <label for="university">University</label>
            <input type="text" name="university" id="university"><br>

            <label for="photo">Photo</label>
            <input type="file" name="photo" id="photo" accept="image/png"><br>

            <label for="transcripts">Transcripts</label>
            <input type="file" name="transcripts" id="transcripts" accept="image/
    png"> <br>

            <label for="message">Message</label>
            <textarea name="message" id="message" cols="40" rows="15" placeholder=
    "Type your message here"></textarea>
<br>
            <input type="submit" value="Submit">
            <input type="reset" value="Reset">
            <input type="button" value="Ok">
        </fieldset>
    </form>
</body>
</html>
```

4 (Q No.2B) How can you map images in HTML? Does a hyperlink apply to text only? Explain the logic behind it and its types in detail.

To map an image in HTML, you can use the <map> and <area> elements to define clickable regions on the image. Here's an example:

```
    <img src="https://fakeimg.pl/300/" alt="My Image" usemap="#myMap">

    <map name="myMap">
    <area shape="rect" coords="0,0,100,100" href="page1.html" alt="Page 1">
    <area shape="rect" coords="100,0,200,100" href="page2.html" alt="Page 2">
```

```
  <area shape="circle" coords="150,150,50" href="page3.html" alt="Page 3">
  </map>
```

Listing 1: Image map

When the user clicks on one of the mapped areas, the browser will navigate to the URL specified in the ”href” attribute. This allows you to create interactive images that can act as navigation menus, image maps, and more.

No, hyperlinks in HTML do not apply to text only. We can create hyperlinks for any element that can have a clickable area, including images, buttons, and even entire divs.

A hyperlink is created using an ¡a¿ tag, which stands for ”anchor.” The <a> tag requires a ”href” attribute, which specifies the URL that the link should point to. We can also include text or other elements within the <a> tag, which will serve as the clickable area for the hyperlink.

Here’s an example of how to create a hyperlink for text in HTML:

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

For email links also we can use the hyperlink. Email links are created using the mailto: protocol, like this:

```
  <a href="mailto:example@example.com">Send an email to Example</a>
```

For images we can use hyperlink like this:

```
  <a href="www.google.com"><img src="./google.png" alt="Click here to go to
 google"> </a>
```

Any clickable content inside the <a> tag works for hyperlink.

---

5 (Q No.3A) What are the overriding rule in CSS? Explain different levels of stylesheets with their hierarchy of implementation.

In CSS, the overriding rule is used to determine which style declarations should be applied to an element when there are multiple rules that could potentially apply.

The overriding rule in CSS can be summarized as follows:

1. Styles defined using the !important keyword take precedence over all other styles, regardless of where they are defined in the CSS file.

2. Styles defined inline (i.e. using the style attribute in an HTML element) take precedence over styles defined in an external CSS file or in the <head> section of the HTML document.

3. Styles defined closer to the element in the HTML document take precedence over styles defined further away. For example, styles defined in an ID selector take precedence over styles defined in a class selector, and styles defined in a class selector take precedence over styles defined in a tag selector.

4. Styles defined with more specific selectors take precedence over styles defined with less specific selectors. For example, a style defined with a selector of #my-div p (which targets all <p> elements within an element with the ID of my-div) would take precedence over a style defined with a selector of p (which targets all <p> elements on the page).

5. Styles defined later in the CSS file take precedence over styles defined earlier in the file, provided that all other factors are equal.

There are 3 levels of stylesheets in CSS:

1. Inline Styles: These styles are applied directly to an HTML element using the style attribute. Inline styles have the highest specificity, meaning they override any other styles applied to the same element.

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

```

Listing 2: Inline Style

2. Internal Styles: These styles are defined within the head section of an HTML document using the style tag. Internal styles have a higher specificity than external styles, but lower than inline styles.

```

        <head>
    <style>
        p {
        color: blue;
        }
    </style>
    </head>
    <body>
    <p>This text is blue.</p>
    </body>

```

Listing 3: Internal Style

3. External Styles: These styles are defined in an external CSS file and are linked to an HTML document using the link tag. External styles have the lowest specificity, but they are also the most reusable and easier to maintain.

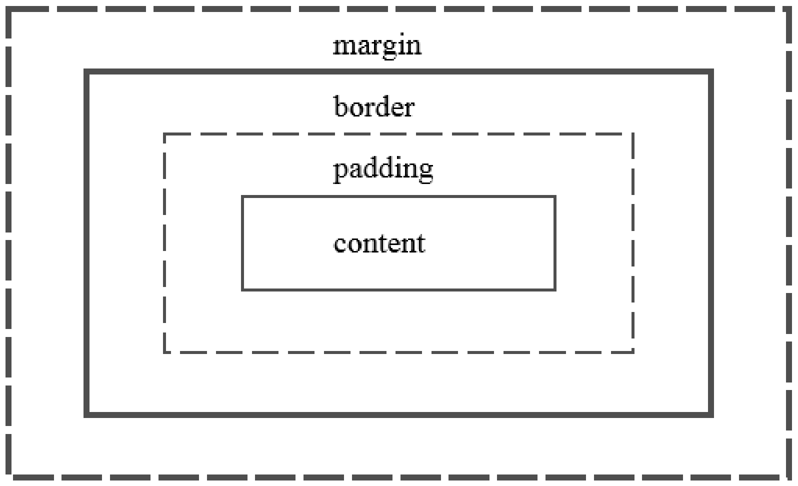
```

            <head>
        <link rel="stylesheet" href="style.css">
        </head>
        <body>
        <p class="red-text">This text is red.</p>
        </body>

```

Listing 4: External Style

---

6 (Q No.3B) Design a box model. What will be the actual height and width of content if padding=margin=border=height=width of content = 20px?

The CSS box model is a container that contains multiple properties including borders, margin, padding, and the content itself. It is used to create the design and layout of web pages. It can be used as a toolkit for customizing the layout of different elements. The web browser renders every element as a rectangular box according to the CSS box model. Box-Model has multiple properties in CSS.



If the padding, margin, border, height, and width of an element's content are all set to 20 pixels, then the actual height and width of the content will be 0 pixels.

This is because the total size of the element would be calculated as follows:

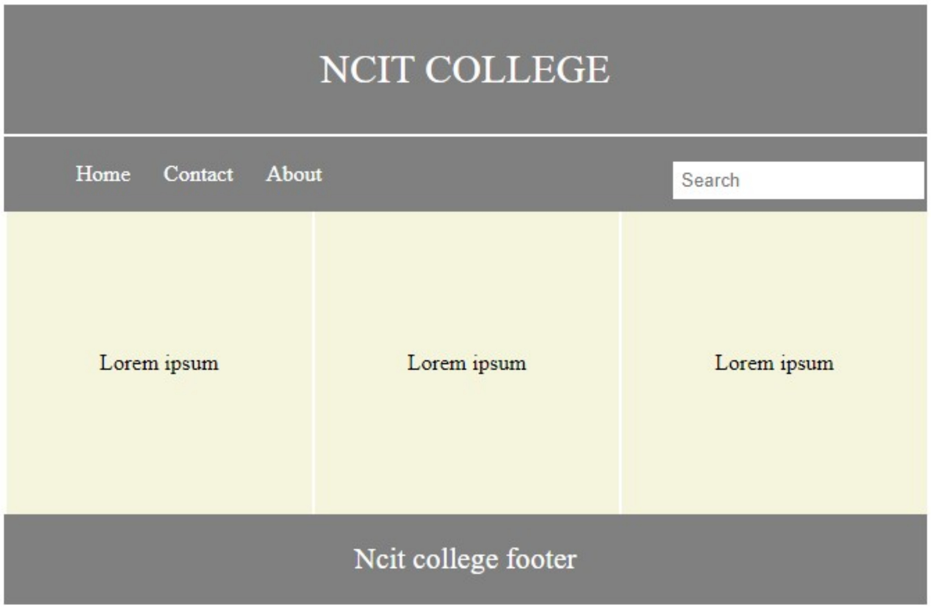
Total size = height + padding + border + margin

Total size = 20px + 20px + 20px + 20px

Total size = 80px

Since the total size is greater than the height and width of the content (which are both set to 20 pixels), there is no space left over for the content itself. As a result, the content will be completely hidden and will not be visible on the page.

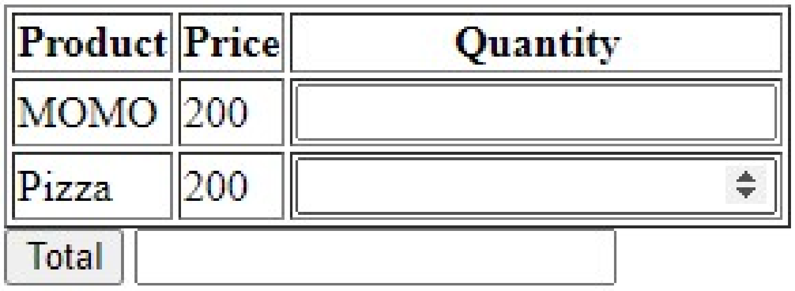
7 (Q No.C) Write HTML and CSS code for the following O/P



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

header {
    padding: 30px;
    background-color: gray;
    text-align: center;
    font-size: 22pt;
    color: white;
}

nav {
    margin-top: 2px;
    background-color: gray;
    padding: 2px;
    display: block;
}

li {
    list-style-type: none;
    display: inline;
    margin: 10px;
}
li a {
    text-decoration: none;
    color: white;
}

li input[type=search] {
    float: right;
    padding: 6px;
    border: none;
    }

    .wrapper {
        display: flex;
    }

    .box {
        /* height: 200px; */
        width: 100%;
        background-color: beige;
        margin-left: 2px;
        text-align: center;
        padding: 100px 0;
    }

    footer {
        text-align: center;
        background-color: gray;
        padding: 20px;
        color: white;
        font-size: 16pt;
    }
    </style>
</head>
<body>
    <header>NCIT COLLEGE</header>
    <nav>

        <ul>
            <li><a href="#home">Home</a></li>
            <li><a href="#contact">Contact</a></li>
            <li><a href="#about">About</a></li>
            <li><input type="search" name="search" id="search" placeholder="Search
    "></li>
    </ul>
    </nav>

    <section class="wrapper">
        <div class="box box1"> Lorem ipsum </div>
        <div class="box box2"> Lorem ipsum </div>
        <div class="box box3">Lorem ipsum </div>
    </section>

    <footer>
        Ncit college footer
    </footer>
</body>
</html>
```

Listing 5: Html css for given design

8 (Q No.4A) How events are handled in JavaScript? Write the JS code for the following o/p. When the user provides quantity it gives total.



Product	Price	Quantity
MOMO	200	
Pizza	200	

In JavaScript, events are used to handle user interactions or system events, such as a mouse click or a keypress, and trigger corresponding actions.

To handle events in JavaScript, you can use event listeners. An event listener is a function that is executed in response to an event. Here's an example:

```
    // Get a reference to the element you want to listen for an event on
const button = document.querySelector('button');

// Add an event listener to the element
button.addEventListener('click', function() {
  // Do something when the button is clicked
  alert("Hello World!!!");
});
In this example, we get a reference to a button element using document.
    querySelector(), and then add an event listener to it using the
    addEventListener() method. We pass in two arguments to addEventListener(): the
    type of event we want to listen for (in this case, 'click'), and the function
    we want to execute when the event occurs.

When the button is clicked, the function passed as the second argument to
    addEventListener() will be executed. Inside this function, we have alert
    function which will create a pop box when clicked.
```

Here is the JS code for the given scenario to display the total when quantity are given.

```html
<html lang="en">
<head>

    <title>Document</title>
</head>
<body>
    <table border="1">
        <tr>
            <th>Product</th>
            <th>Price</th>
            <th>Quantity</th>
        </tr>
        <tr>
            <td>MOMO</td>
            <td id="momoprice">200</td>
            <td><input type="number" name="momo" id="momo" onkeyup="total(event)"></td>

        </tr>
        <tr>
            <td>Pizza</td>
            <td id="pizzaprice">200</td>
            <td><input type="number" name="pizza" id="pizza" onkeyup="total(event)"></td>
        </tr>
    </table>

    <button>Total</button> <input type="text" name="total" id="total" readonly>


    <script>


        function total(event) {
            let momo = document.getElementById("momo").value
            let pizza = document.getElementById("pizza").value
            let momoPrice = document.getElementById("momoprice").innerText
            let pizzaPrice = document.getElementById("pizzaprice").innerText
            let total = document.getElementById("total")

            if(momo == ""){
                momo = 0;
            }
            if(pizza == ""){
                pizza = 0;
            }

            console.log(momo);
            console.log(pizza);
            console.log(momoPrice);
            console.log(pizzaPrice);

            let totalPrice =( parseInt(momo) * parseInt(momoPrice)) + (parseInt(pizza) * parseInt(pizzaPrice))

            console.log(totalPrice);
            total.value = totalPrice
        }
    </script>
</body>
</html>
```

Listing 6: JS code to display total

Here we have used onKeyUp function to handle the event, when key is pressed the event is triggered which will call the total() function in the JS, where we had created reference to the both quantity and price field, and then we checked if value is null then set to 0 for default value, then after converting to integer we calculated the total value then displaye in the total box.

9 (Q No.4B) What do you mean by DOM in JavaScript? Differences between DOM 0 and DOM 1 and DOM 2 methods with example.

In JavaScript, the DOM (Document Object Model) is an API that provides a way for JavaScript code to interact with HTML and XML documents as a tree of objects. The DOM represents the structure of a document as a tree of nodes and provides methods and properties for manipulating those nodes.

The differences between the DOM: DOM Level 0, DOM Level 1, and DOM Level 2 are:

- DOM Level 0: This refers to the original, basic DOM API that was built into early versions of web browsers. It includes a few global objects and methods, such as document, window, and alert(), but doesn't provide a formalized API for working with the document tree. Instead, it allows directly manipulate the properties of the document and its elements using JavaScript. Example of using DOM Level 0:

```
        // Accessing the document object
        const doc = document;

        // Accessing an element and changing its text content
        const heading = document.getElementById('heading');
        heading.textContent = 'New Heading';

```

- DOM Level 1: This introduced a formalized API for working with the document tree, including methods for accessing and manipulating elements and their attributes, as well as event handling. This version of the DOM is widely supported in modern browsers. Example of using DOM Level 1:

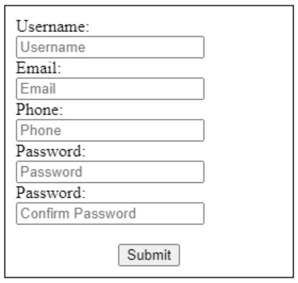
```
        // Accessing an element by ID and changing its text content
        const heading = document.getElementById('heading');
        heading.textContent = 'New Heading';

        // Adding an event listener to a button element
        const button = document.getElementById('myButton');
        button.addEventListener('click', function() {
        console.log('Button clicked!');
        });


```

- DOM Level 2: This added additional features to the DOM, such as support for CSS styling and more advanced event handling. This version of the DOM is also widely supported in modern browsers. Example of using DOM Level 2:

```
        // Accessing an element's style properties
        const box = document.getElementById('myBox');
        box.style.backgroundColor = 'red';
        box.style.width = '100px';
        box.style.height = '100px';

        // Adding a mouseover event listener to an element
        const button = document.getElementById('myButton');
        button.addEventListener('mouseover', function() {
```

```
        this.style.backgroundColor = 'blue';
        });


```

10 (Q No.5A) Create a registration form as given. Also validates it with JavaScript. (All input should be fulfilled else *required message will display, the name must be of 8 characters, email should in the proper format. the phone must be of 10 digits, password include only number and must be of 8 digits, and starting and ending with 0, Confirm password and Password must match).



Here is the requied JS code to validate the given requirements:

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

    <style>
        .register-form {
            border: 1px solid black;
            width: 250px;
            padding: 10px;
        }
        .error {
            color: red;
        }
    </style>
</head>
<body>
<div class="register-form">
    <form id="registrationForm">
        <label for="username">Username:</label><br>
        <input type="text" id="username" name="username" minlength="8" placeholder
    ="Username">
```

```
      <br>
      <label for="email">Email:</label><br>
      <input type="email" id="email" name="email" placeholder="Email">
      <br>
      <label for="phone">Phone:</label><br>
      <input type="tel" id="phone" name="phone" minlength="10" maxlength="10" placeholder="Phone">
      <br>
      <label for="password">Password:</label><br>
      <input type="password" id="password" name="password" placeholder="Password" pattern="^0\d{6}0$">
      <br>
      <label for="confirmPassword">Password:</label><br>

      <input type="password" id="confirmPassword" name="confirmPassword" placeholder="Confirm Password">
      <br>
      <br>
      <div style="text-align: center;">
          <input type="submit" value="Submit">
      </div>
    </form>

    <div id="errorMessages" class="error"></div>


    <script>
      const form = document.getElementById('registrationForm');
      const errorMessages = document.getElementById('errorMessages');

      form.addEventListener('submit', function(event) {
      event.preventDefault();
      errorMessages.innerHTML = '';

      const username = document.getElementById('username').value;
      const email = document.getElementById('email').value;
      const phone = document.getElementById('phone').value;
      const password = document.getElementById('password').value;
      const confirmPassword = document.getElementById('confirmPassword').value;

        // regular expressions for validation
      const usernameRegex = /^[a-zA-Z0-9_]{8}$/;
      const emailRegex = /^\S+@\S+\.\S+$/;
      const phoneRegex = /^\d{10}$/;
      const passwordRegex = /^0\d{6}0$/;

      let errorMsg = "";

      // check if all inputs are filled
      if (!username || !email || !phone || !password || !confirmPassword) {
          errorMsg += '<p>*All fields are required.</p>'
          errorMessages.innerHTML = errorMsg;
          return;
      }

      let isValid = true;

      if (!usernameRegex.test(username)) {
```

By: Sanjaya (Bir Bikram) Shrestha

```
            errorMsg += "Username must be 8 characters long and can only contain alphanumeric characters and underscore (_).<br>";
            isValid = false;
        }

        // check if email is valid
        if (!emailRegex.test(email)) {
            errorMsg += "Email must be in proper format.<br>";
            isValid = false;

        }

        // check if phone number is valid
        if (!phoneRegex.test(phone)) {
            errorMsg += "Phone number must be 10 digits long.<br>";
            isValid = false;

        }

        // check if password is valid
        if (!passwordRegex.test(password)) {
            errorMsg += "Password must be 8 digits long and start and end with 0.<br>";
            isValid = false;

        }

        // check if password and confirm password match
        if (password !== confirmPassword) {
            errorMsg += "Password and confirm password must match.<br>";
            isValid = false;

        }

        if (isValid) {
            // Form is valid, do something with the data
            console.log('Form submitted successfully.');
            form.reset();
        } else {
            errorMessages.innerHTML = errorMsg;
        }
    });
      </script>
</div>

</body>
</html>
```

Listing 7: Html Js code for validation

11 (Q No.5B) What are session and cookies? Mention any five difference between them with an example. An example must include all the function of the session and cookies. Also writes a PHP session code that helps detect no of website visit.

Session and cookies are two ways to store data between HTTP requests in PHP. Here are some key differences:

1. **Storage Location:** Session data is stored on the server, while cookie data is stored on the client (in the browser).
2. **Data Size:** Cookies have a maximum size limit of around 4 KB, while session data can be much larger.
3. **Security:** Session data is generally considered more secure than cookies, as it is stored on the server and cannot be easily tampered with by the client.
4. **Expiration:** Cookies can have an expiration date, while session data is destroyed when the user closes their browser or logs out.
5. **Accessibility:** Cookies can be accessed by both the client-side scripts and server-side scripts, while session data can only be accessed by server-side scripts.
6. Cookies can be set using the setcookie() function. This function takes several parameters including the cookie name, value, expiration time, and path.

```
        setcookie('username', 'john_doe', time() + 3600, '/');

```

For Retrieving data from cookie we use following code:

```
        // Check if the 'username' cookie is set and retrieve its value
        if (isset($_COOKIE['username'])) {
        $username = $_COOKIE['username'];
        echo "Welcome back, $username!";
        } else {
        echo "Please log in to continue.";
        }

```

For starting session and setting values in it we use following code:

```
        // Start the session
        session_start();

        // Check if the 'username' session variable is set and retrieve its
    value
        if (isset($_SESSION['username'])) {
        $username = $_SESSION['username'];
        echo "Welcome back, $username!";
        } else {
        echo "Please log in to continue.";
        }


```

For deleting cookie and session we use following code: Here to remove cookie we reset cookie time to 1 hour before, using -3600 which expires the cookie. And for session we use session_destroy() function.

```
        // removing data from session
        unset($_SESSION['username']);

        // for removing cookie
        setcookie('username', 'john_doe', time() - 3600, '/');

        // for removing session
        session_destroy();


```

---

12 (Q No.6A) Explain $_SERVER, $_POST, $_GET, $_REQUEST. Also, validate the form of question 5a using server-side scripting.

In PHP, $_SERVER, $_POST, $_GET, and $_REQUEST are predefined variables used to handle different types of user input.

- $_SERVER The $_SERVER variable contains information about the web server and the current request. It is an array that contains various properties like the current script name, the request method (e.g. GET, POST), and the server address.

- $_POST The $_POST variable contains data that has been submitted via an HTTP POST request. It is an associative array where the keys are the input names and the values are the input values.

  Here's an example of how to use $_POST to retrieve a form input value:

- $_GET

  The $_GET variable contains data that has been submitted via an HTTP GET request. It is also an associative array where the keys are the input names and the values are the input values.

- $_REQUEST

  The $_REQUEST variable contains data that has been submitted via either an HTTP GET or POST request. It is also an associative array where the keys are the input names and the values are the input values.

  Here's example for all type of request:

```
        // get the current script name
        $script_name = $_SERVER['SCRIPT_NAME'];
        // when form is submittes using post request
        $username = $_POST['username'];

        // when form is submittes using get request
        $username = $_GET['username'];

        // to get data sent from any form request eg: GET and POST
        $username = $_REQUEST['username'];


```

Here is the PHP code to validate the given form:

```php
<?php
// define variables and set to empty values
$username = $email = $phone = $password = $confirm_password = "";
$username_err = $email_err = $phone_err = $password_err = $confirm_password_err =
    "";

// check if form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {

  // validate username
  if (empty($_POST["username"])) {
    $username_err = "Username is required";
  } else {
    $username = test_input($_POST["username"]);
    if (strlen($username) != 8) {
      $username_err = "Username must be exactly 8 characters";
    }
  }

  // validate email
  if (empty($_POST["email"])) {
    $email_err = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $email_err = "Invalid email format";
    }
  }

  // validate phone
  if (empty($_POST["phone"])) {
    $phone_err = "Phone is required";
  } else {
    $phone = test_input($_POST["phone"]);
    if (!preg_match("/^[0-9]{10}$/", $phone)) {
      $phone_err = "Phone must be 10 digits";
    }
  }

  // validate password
  if (empty($_POST["password"])) {
    $password_err = "Password is required";
  } else {
    $password = test_input($_POST["password"]);
    if (!preg_match("/^0[0-9]{6}0$/", $password)) {
      $password_err = "Password must be 8 digits and start and end with 0";
    }
  }

  // validate confirm password
  if (empty($_POST["confirm_password"])) {
    $confirm_password_err = "Confirm password is required";
  } else {
    $confirm_password = test_input($_POST["confirm_password"]);
    if ($password !== $confirm_password) {
      $confirm_password_err = "Passwords do not match";
    }
  }

  // if there are no errors, redirect to success page
  if (empty($username_err) && empty($email_err) && empty($phone_err) && empty(
   $password_err) && empty($confirm_password_err)) {
    header("Location: success.php");
    exit;
  } else {
    header("Location: errors.php");
    exit;
    exit;
  }
}

// helper function to sanitize input values
function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

13 (Q No.6B) Develop a login form that includes Username and Password (Username="admin" Password="admin" already presents in your database). Write a PHP and MYSQL first to verify log in credentials. Second when login is successful it navigates to the form in question 5. When you fill out the form it should dispaly the information in the table structure. (Show database connection and assume database name and table name yourself)

Here is the source code to check login and redirect to form.php file:

```
    <?php
  // Database connection
  $servername = "localhost";
  $username = "root";
  $password = "";
  $dbname = "mydatabase";

  $conn = new mysqli($servername, $username, $password, $dbname);

  if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
  }
```

```

// Check if username and password are correct
  $username = $_POST['username'];
$password = $_POST['password'];

// Query database for matching user
$sql = "SELECT * FROM users WHERE username = '$username' AND password = '
 $password'";
$result = $conn->query($sql);

if ($result->num_rows == 1) {
  // If login is successful, display a success message and redirect to form
  echo "Login successful";
  header("Location: form.php");
  exit();
} else {
  // If login fails, display an error message
  echo "Invalid username or password";
}
?>
```

On successful login we have redirected to the form.php now, when form.php is submitted and validates we then redirect to the succes.php Here is the success.php code where we display all the data from table,

Here is the form.php extra code that will insert data to database,

```

  // Insert data into table
  $sql = "INSERT INTO users (username, email, phone, password) VALUES ('
 $username', '$email', '$phone', '$password')";

  if ($conn->query($sql) === TRUE) {
      // If data is successfully inserted, display a success message
      echo "Data successfully submitted";
      header("Location : success.php") ;
  } else {
      // If an error occurs, display an error message
      echo "Error: " . $sql . "<br>" . $conn->error;
  }

  $conn->close();
```

After redirecting to success.php we now display data in table format, reading from database table.

```
  <?php
// Database connection
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "mydatabase";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// Select data from table
```

```
  $sql = "SELECT * FROM users";
  $result = $conn->query($sql);
?>

<!DOCTYPE html>
<html>
<head>
  <title>Table</title>
</head>
<body>
  <h2>Table</h2>
  <table>
    <tr>
      <th>Name</th>
      <th>Email</th>
      <th>Phone</th>
    </tr>
    <?php
      // Display data in table rows
      if ($result->num_rows > 0) {
        while($row = $result->fetch_assoc()) {
          echo "<tr><td>" . $row["username"] . "</td><td>" . $row["email"] . "</td
><td>" . $row["phone"] . "</td></tr>";
        }
      } else {
        echo "0 results";
      }
    ?>
  </table>
</body>
</html>

<?php
  $conn->close();
?>
```

14 (Q No.7A) Arrays in PHP

In PHP, there are three types of arrays:

1. **Indexed arrays:** An indexed array stores a collection of values, each of which is assigned a numeric index. The index starts at 0 and increases by 1 for each element in the array. Indexed arrays can be created using the array() function or using the shorthand [] syntax. Here's an example:

```
        $names = array("ram", "shyam", "hari");
        // or
        $names = ["ram", "shyam", "hari"];

```

2. **Associative arrays:** An associative array stores a collection of key-value pairs, where each key is a string and each value can be of any type. Associative arrays can be created using the

$$array()$$

function or using the shorthand [] syntax, but the key-value pairs must be explicitly defined using the => operator. Here's an example:

```php
        $person = array("name" => "John", "age" => 30, "email" => "
    john@example.com");
        // or
        $person = ["name" => "John", "age" => 30, "email" => "john@example.com
    "];


```

3. **Multi dimensional array:** Multidimensional array is an array that contains one or more arrays as its elements. Each element of a multidimensional array can be an array itself, which can contain further arrays as its elements.

   Multidimensional arrays are useful for representing complex data structures, such as tables or matrices. They can be indexed using multiple sets of keys, which allows for flexible and efficient data access.

```php
        $data = array(
            array("John", "Doe", 25),
            array("Jane", "Smith", 30),
            array("Bob", "Johnson", 40)
        );

```

15 (Q No.7B) Key press events in JS

In JavaScript, you can use the keypress event to detect when a key is pressed on the keyboard. Here's an example:

```html
    <!DOCTYPE html>
<html>
<head>
  <title>Keypress Event Example</title>
</head>
<body>
  <input type="text" id="myInput">

  <script>
    // Get the input element
    var input = document.getElementById("myInput");

    // Add an event listener for the keypress event
    input.addEventListener("keypress", function(event) {
      // Get the key code of the pressed key
      var keyCode = event.key;

      // Check if the key is the Enter key (keyCode 13)
      if (keyCode === "Enter") {
        // Do something when the Enter key is pressed
        alert("Enter key pressed!");
      }
    });
  </script>
```

```
</body>
</html>
In this example, an input element is added to the HTML document with an id of
    myInput. A JavaScript event listener is then added to the input element to
    listen for the keypress event.

When a key is pressed, the event listener function is called with an event
    parameter. The function first checks whether the keyCode property of the event
    is available. It then checks whether the key pressed is the Enter key. If the
    Enter key is pressed, the function displays an alert message.
```

16 (Q No.7B) DOM Tree

The DOM (Document Object Model) hierarchy in JavaScript refers to the structure of HTML elements and their relationships with each other. The DOM represents an HTML document as a tree-like structure, where each node in the tree represents an element in the document. Here's an example of a simple HTML document and its corresponding DOM hierarchy:

```html
    <!DOCTYPE html>
    <html>
      <head>
        <title>My Page</title>
      </head>
      <body>
        <h1 id="myHeading">Welcome to my page!</h1>
        <p>This is a paragraph.</p>
        <ul>
          <li>List item 1</li>
          <li>List item 2</li>
          <li>List item 3</li>
        </ul>

        <script>
          // Retrieve the h1 element by its ID
          var heading = document.getElementById('myHeading');

          // Change the text content of the h1 element
          heading.textContent = 'Hello, world!';
        </script>
      </body>
    </html>
```

In this example, the root node of the DOM hierarchy is the html element. The head and body elements are child nodes of the html element, and the title, h1, p, and ul elements are child nodes of the head and body elements. The li elements are child nodes of the ul element.

In JavaScript, we can use the DOM API to manipulate the DOM hierarchy. To retrieve the h1 element and change its text content: we use the getElementById() method of the document object to retrieve the h1 element by its ID. We then use the textContent property to change the text content of the h1 element.