# Web Technology Solution
# 2020 Spring

Sanjaya (Bir Bikram) Shrestha

February 25, 2023

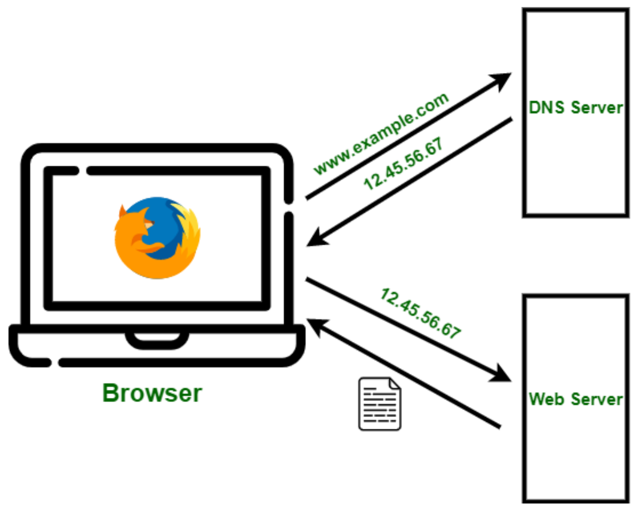
## Contents

## 1 (Q No.1 A) What happens if you enter https://pu.edu.np into your browsers? Describe the procedures behind the scenes. Describe along with its block diagram.

When we type the https://pu.edu.np following sequence of events will happen behind the scene.

1. Browser will initiates a request to the Domain Name System (DNS) to resolve the domain name ”pu.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          |
    +------------------+  +------------------+
```



## 2 (Q No.2 Differentiate between inline, internal and external CSS with an example? Int he code snippet below, what would be the color of both elements 1 and 2? Also explain why?)

```
style.min.css
#about p.element-2{color:red;}
.about p.element-2{color:black;}

index.html

----------
<div id="about" class="about">
    <p class="element-2"> <!-- Element 1 -->
        All right reserved
    </p>
    <p id="element-2"> <!-- Element 2 -->
        I love Web Technology
    </p>

</div>
```

CSS (Cascading Style Sheets) is a styling language used to apply styles to HTML elements. There are three ways to add CSS to an HTML document: inline, internal, and external.

- **Inline CSS:** Inline CSS is when you apply styles directly to an HTML element using the ”style” attribute. Here’s an example:

```
        <h1 style="color: blue;">This is an inline heading</h1>

```

  In this example, the ”style” attribute is used to set the color of the h1 element to blue. Inline CSS is typically used for quick styling fixes or for styles that only apply to a single element.

- **Internal CSS:** Internal CSS is when you define styles within the <head> section of an HTML document, using the <style> tag. Here’s an example:

```
        <head>
        <style>
            h1 {
            color: blue;
            }
        </style>
        </head>
        <body>
        <h1>This is an internal heading</h1>
        </body>

```

In this example, the "style" element is used to define a style for the h1 element. Internal CSS is typically used for styling multiple elements on a single page.

- **External CSS:** External CSS is when you define styles in a separate file and link to it in the <head> section of an HTML document. Here's an example:

```
        <head>
        <link rel="stylesheet" type="text/css" href="styles.css">
        </head>
        <body>
        <h1>This is an external heading</h1>
        </body>


```

In this example, the "link" element is used to link to an external CSS file called "styles.css", which contains the styles for the h1 element. External CSS is typically used for styling multiple pages across a website.

In the given code,
the color of element 1 will be red and
the color of element 2 will be black.

This is because of the CSS selector specificity. The selector #about p.element-2 has a higher specificity than .about p.element-2, because it includes an ID selector #about which has a higher specificity than a class selector .about. So, for the element with class element-2 inside the #about div, the first selector will be applied, which sets the color to red.

On the other hand, the element 2 has a id of element-2 but it doesn't have an class of element-2, so css is specified to work for p element with id element-2 so the default style will be applied so the color will be set to black.

---

## 3 (Q No.3 Write a program to display the following output.)

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

    <title>Document</title>
</head>
<body>
    <table border="1">
        <tr>
            <th>No.</th>
            <th>Full Name</th>
            <th>Club</th>
            <th>Salary</th>
            <th>Type</th>
        </tr>
        <tr>
            <td>1</td>
            <td>Lionel Messi</td>
            <td>FC Barcelona</td>
            <td>130 million EUR</td>
            <td rowspan="4">Football Player</td>
        </tr>
        <tr>
            <td>2</td>
            <td>Cristiano Ronaldo</td>
            <td>Juventus</td>
            <td>96.4 million EUR</td>
        </tr>
        <tr>
            <td>3</td>
            <td>Neymar</td>
            <td>Paris Saint</td>
            <td>92.8 million EUR</td>
        </tr>
        <tr>
            <td>4</td>
            <td>Paul Pogba</td>
            <td>Manchester United</td>
            <td>29.2 million EUR</td>
        </tr>
        <tr>
            <td colspan="3"  style="text-align: center;"><b>Total Expense:</b></td
    >
            <td>348 million EUR</td>
        </tr>
    </table>
</body>
</html>
```

Listing 1: Table in Html

## 4 (Q No.4 ”JavaScript programs the behaviour of the webpage”. Explain this statement with suitabel example. Employ DOM event model in JavaScript to illustrate the dragging and dropping of element with suitabel example.)

The statement ”JavaScript programs the behavior of the webpage” means that JavaScript is used to add interactivity and dynamic functionality to a webpage. Here’s an example to explain this:

Suppose we have a webpage with a button that when clicked, displays a message. We can achieve this using JavaScript.

First, we need to add an event listener to the button so that it listens for when the button is clicked. We can do this using the addEventListener() method in JavaScript:

```
    <button id="myButton">Click me</button>

    <script>
    const button = document.getElementById("myButton");

    button.addEventListener("click", function() {
        alert("Hello, world!");
    });
</script>
```

```
<!DOCTYPE HTML>
<html>
<head>
<style>
#div1 {
  width: 350px;
  height: 70px;
  padding: 10px;
  border: 1px solid #aaaaaa;
}
</style>
<script>
function allowDrop(ev) {
  ev.preventDefault();
}

function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<p>Drag the W3Schools image into the rectangle:</p>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
<br>
<img id="drag1" src="https://fakeimg.pl/300/" draggable="true" ondragstart="drag(
    event)" width="336" height="69">

</body>
</html>
```

Listing 2: Drag and Drop

To make an element draggable we set the draggable attribute to true of the element. The ondragstart attribute calls a function, drag(event), that specifies what data to be dragged.

The dataTransfer.setData() method sets the data type and the value of the dragged data: The ondragover event specifies where the dragged data can be dropped.

## 5 (Q No.5A What are the different types of Array in PHP? Explain the details with syntax(Use the name of your friend as example))

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:

```php
        $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)
        );

```

## 6 (Q No.5B What role do cookies and session play in allowing web servers to store stateful data? Give example to illustrate your point.

Cookies and sessions are both used to allow web servers to store stateful data, which means data that persists across multiple requests.

Cookies are small text files that are stored on the client-side (i.e. on the user's browser). They are typically used to store user preferences, login information, shopping cart items, and other small amounts of data that need to persist between visits to a website. Cookies are sent to the server with every request, so the server can read and modify the contents of cookies to store and retrieve stateful data.

Sessions, on the other hand, are stored on the server-side. When a user visits a website, the server creates a session for that user and assigns a unique identifier (usually a session ID) to that session. The session ID is then sent to the client in the form of a cookie, so that the server can identify the user in subsequent requests. The server can then store stateful data related to that user in the session, and retrieve it as needed.

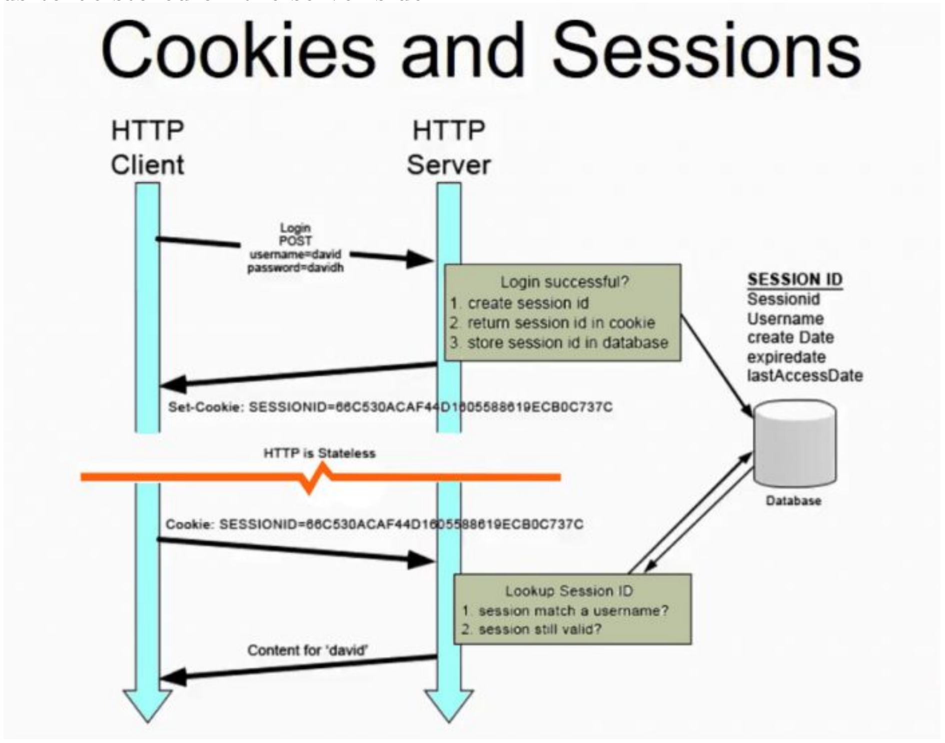
Here's an example to illustrate how cookies and sessions can be used to store stateful data:

Suppose we have an online store where users can add items to their shopping cart. When a user adds an item to their cart, we want to store that information so that it persists even if the user navigates away from the site and comes back later.

We can use cookies to store the shopping cart items on the client-side. Each time the user adds an item to their cart, we can update a cookie with the item details. Then, when the user returns to the site, we can read the contents of the cookie and display the items in their cart.

However, cookies have limitations in terms of storage capacity and security. To store sensitive data like payment information, we need a more secure and reliable way to store stateful data. That's where sessions come in. We can create a session for each user and store their shopping cart items in the session. This way, the data is stored on the server-side and is more secure than using cookies. We can also store larger amounts of data in a session than we can in a cookie.

In summary, cookies and sessions are both used to store stateful data, but they have different properties and use cases. Cookies are typically used for small amounts of data that need to persist on the client-side, while sessions are used for larger amounts of data and more sensitive data that needs to be stored on the server-side.


Cookies and Sessions
HTTP Client
HTTP Server
Login POST username=david password=davidh
Login successful?
1. create session id
2. return session id in cookie
3. store session id in database
SESSION ID
Sessionid
Username
create Date
expiredate
lastAccessDate
Set-Cookie: SESSIONID=66C530ACAF44D1605588619ECB0C737C
HTTP is Stateless
Database
Cookie: SESSIONID=66C530ACAF44D1605588619ECB0C737C
Lookup Session ID
1. session match a username?
2. session still valid?
Content for 'david'

## 7 (Q No.6 Prepare a login form with input field username/email and password and also the option for remembering the login session. Write PHP MySQL code to establish the database connection and verify the user and successfully login and establish login session for valid users and show error messages if invalid. Use the client and server side validation, along with pattern matching implemented to allow only authorized user.

Here is the source code for given scenario.

```
<!DOCTYPE html>
<html>
<head>
    <title>Login Form</title>
    <script>
        function validateForm() {
            var email = document.forms["loginForm"]["email"].value;
            var password = document.forms["loginForm"]["password"].value;

            // Check if email and password fields are empty
            if (email == "" || password == "") {
                alert("Please fill in all the required fields.");
                return false;
            }

            // Check if email is in valid format
            var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
            if (!emailRegex.test(email)) {
                alert("Please enter a valid email address.");
                return false;
            }

            // Check if password is at least 8 characters long
            if (password.length < 8) {
                alert("Password should be at least 8 characters long.");
                return false;
            }

            return true;
        }
    </script>
</head>
<body>
    <form name="loginForm" method="post" action="login.php" onsubmit="return
   validateForm()">
        <label>Email:</label><br>
        <input type="email" name="email"><br>
        <label>Password:</label><br>
        <input type="password" name="password"><br>
        <input type="checkbox" name="remember_me" value="1">Remember Me<br>
        <input type="submit" value="Login">
    </form>
</body>
```

```
</html>
```

Listing 3: Login Page: index.html

```php
<?php
session_start();
// Establish database connection
$conn = mysqli_connect("localhost", "username", "password", "database_name");

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Check if form is submitted
if (isset($_POST['username']) && isset($_POST['password'])) {
    // Retrieve the entered username and password
    $username = mysqli_real_escape_string($conn, $_POST['username']);
    $password = mysqli_real_escape_string($conn, $_POST['password']);

    // Check if the user wants to remember the login session
    if (isset($_POST['remember'])) {
        // Set cookie to remember the login session for 30 days
        setcookie("username", $username, time() + (86400 * 30), "/");
        setcookie("password", $password, time() + (86400 * 30), "/");
    }

    // Query to verify the user
    $query = "SELECT * FROM users WHERE (username='$username' OR email='$username
    ') AND password='$password'";
    $result = mysqli_query($conn, $query);

    // Check if the query returns any rows
    if (mysqli_num_rows($result) == 1) {
        // User is valid, create a session and redirect to homepage
        $_SESSION['username'] = $username;
        header("Location: homepage.php");
    } else {
        // User is invalid, display an error message
        $error_message = "Invalid login credentials";
    }
}

// Close database connection
mysqli_close($conn);
?>
```

Listing 4: Login Page: login.php

In the above code, we start the session and establish a database connection using mysqli_connect(). We then check if the login form is submitted by checking if the $_POST array contains values for username and password fields. We also check if the user has checked the "remember me" checkbox, and if so, we set cookies to remember the login session for 30 days.

We then use mysqli_real_escape_string() to sanitize the entered username and password, and execute a query to retrieve the user with the given credentials from the database. If the query returns a row, we create a session variable named 'username' and set it to the entered username, and then redirect the user to the homepage. If the query doesn't return any rows, we display an error message.

Finally, we close the database connection using mysqli_close(). Here We have some basic validation in client side using JavaScript which will check for null value, password should be at least 8 character and email patter using regular expression. Once every validaion is passed then we submit form to the server, login.php.