1. Setting and Retrieving a User Preference

Objective: Create a PHP script that allows the user to select their preferred background color from a dropdown list. Store this preference in a cookie and apply the background color whenever the user visits the page.

Steps:

- Create a form with a dropdown list that has at least 3 color options (e.g., Red, Blue, Green).
- When the form is submitted, save the selected color in a cookie.
- On page load, check if the color cookie is set, and if so, apply the selected color as the background color of the page.

---

2. Visit Counter with Cookies

Objective: Create a script that counts how many times the user has visited the page using cookies.

Steps:

- When the user visits the page for the first time, set a cookie with the value of 1.
- On each subsequent visit, update the cookie to increment its value.
- Display the total number of visits to the user.

---

3. Remember Username

Objective: Create a login form that remembers the username entered using a cookie.

Steps:

- Create a simple login form with a username input.
- When the form is submitted, set a cookie to store the username.
- On subsequent visits, automatically fill in the username input field with the value stored in the cookie.

---

4. Cookie Expiry Experiment

Objective: Set a cookie with a specific expiration time and test its behavior.

Steps:

- Create a script that sets a cookie with an expiration time of 1 minute.
- Display the cookie value while it's valid.
- Wait for the cookie to expire (after 1 minute), and then try to access it again to observe what happens when a cookie expires.

---

5. Shopping Cart Simulation

Objective: Simulate a simple shopping cart using cookies to store selected items.

Steps:

- Create a page that lists several products with "Add to Cart" buttons.
- When the user adds an item to the cart, store the product name in a cookie.
- On the cart page, retrieve and display all items added to the cart using cookies.

---

6. Deleting a Cookie

Objective: Create a script where the user can delete a cookie.

Steps:

- Set a cookie with some value (e.g., a user's favorite fruit).
- Display the value of the cookie on the page.
- Add a "Delete Cookie" button that, when clicked, will delete the cookie and notify the user that the cookie has been removed.

---

7. Setting Secure Cookies

Objective: Create a script that sets a secure cookie that can only be accessed over HTTPS and is HTTP-only.

Steps:

- Set a cookie with the following options: `secure` and `httponly`.

- Try to retrieve the cookie using JavaScript to confirm that it is HTTP-only and cannot be accessed from the client-side.

---

8. Multi-page Cookie Sharing

Objective: Create two separate pages (`page1.php` and `page2.php`). On the first page, set a cookie, and on the second page, retrieve and display the cookie value.

Steps:

- In `page1.php`, set a cookie with some value (e.g., a message like "Hello from Page 1").
- In `page2.php`, retrieve and display the value of the cookie set on `page1.php`.
- Make sure both pages are on the same domain or path to share the cookie.

---

9. Cookie Update

Objective: Create a script where the user can update an existing cookie value.

Steps:

- Set a cookie with an initial value (e.g., "User Level: Beginner").
- Provide an input field where the user can update their "User Level" (e.g., to "Intermediate" or "Advanced").
- When the form is submitted, update the cookie with the new value and display the updated value to the user.

---

10. Quiz with Cookies

Objective: Create a small quiz where the user’s score is saved in a cookie.

Steps:

- Create a PHP page with a 3-question multiple-choice quiz.
- Store the user's score in a cookie.
- Display the user's total score every time they visit the quiz page.

1. User Login System Using Sessions

Objective: Create a simple login system where user credentials are stored in a session and the user can access a protected page after logging in.

Steps:

- Create a login form with fields for a username and password.
- When the form is submitted, validate the credentials (you can hardcode a username and password).
- If the credentials are correct, store the username in a session.
- Redirect the user to a protected page that can only be accessed if the session is active (i.e., the user is logged in).
- Provide a "Logout" button that destroys the session.

---

2. Flash Messages with Sessions

Objective: Create a PHP script that uses sessions to display **flash messages** (temporary messages that disappear after a page reload).

Steps:

- When the user submits a form, store a success or error message in a session variable.
- On the next page load, display the message to the user.
- After the message is displayed, remove the session variable so that the message disappears on the next reload.

---

3. Shopping Cart Using Sessions

Objective: Simulate a shopping cart system using sessions to store the items selected by the user.

Steps:

- Create a list of products with an "Add to Cart" button next to each.
- When the user clicks "Add to Cart," store the product details (e.g., name, price) in a session.
- Display a cart page that lists all items added to the session.
- Provide a button to clear the cart (destroy the session).

---

4. Multi-page Form Using Sessions

Objective: Create a multi-page form where user inputs from previous steps are stored in the session until the final step.

Steps:

- Step 1: Collect basic user information (e.g., name, email) and store it in a session.
- Step 2: Collect additional details (e.g., address, phone) and store them in the same session.
- Final Step: Display all the information entered by the user from previous steps and allow the user to submit the data.

---

5. Session Expiration

Objective: Set up a session that expires after a specific time of inactivity and redirect the user to a login page when the session expires.

Steps:

- Set a session timeout (e.g., 5 minutes) based on the last activity time.
- Store the last activity time in the session.
- Check the session time on each page request, and if the user has been inactive for more than 5 minutes, destroy the session and redirect the user to the login page.

---

6. Role-based Access Control Using Sessions

Objective: Create a simple role-based access system where different users have different access levels based on their roles stored in sessions.

Steps:

- When a user logs in, assign them a role (e.g., admin, editor, viewer) and store it in a session.
- Create different pages that are accessible only to users with specific roles.
- If a user tries to access a page they don’t have permission for, display an "Access Denied" message.

---

7. Storing User Preferences Using Sessions

Objective: Create a system where a user can change certain preferences (e.g., theme, language) that are stored in sessions and persist during their session.

Steps:

- Create a form where users can select their preferred theme (e.g., light or dark) and language (e.g., English or Arabic).
- Store these preferences in the session.
- Apply the preferences to the page (e.g., change the theme or language) based on the session data.

---

8. Quiz with Session-based Score Tracking

Objective: Create a multi-question quiz where the user's score is stored in a session as they answer each question.

Steps:

- Display one question at a time, with options to select the correct answer.
- After each question is answered, store whether the answer was correct in the session.
- After all questions are answered, display the total score stored in the session.

---

9. Prevent Session Hijacking

Objective: Create a secure session mechanism that regenerates session IDs to prevent session hijacking.

Steps:

- Upon user login, regenerate the session ID using `session_regenerate_id()`.
- Store user IP or user agent details in the session for additional security checks.
- If the IP address or user agent changes during the session, destroy the session and log the user out.

---

10. Session-based Survey

Objective: Create a simple survey where the user can submit their answers, and the answers are stored in a session.

Steps:

- Display a survey form with multiple questions.
- When the user submits the form, store the answers in a session.
- After the survey is submitted, show a summary of the user's answers stored in the session.

---

1. Handling Form Data Using `$_POST`

Objective: Create a PHP script that processes a form using the `$_POST` superglobal.

Steps:

- Create a form that collects a user's first name, last name, and email.
- When the form is submitted, use the `$_POST` superglobal to retrieve and display the submitted data.
- Make sure to handle basic validation (e.g., required fields).

---

2. Search Query Using `$_GET`

Objective: Create a search form that uses the `$_GET` superglobal to pass the search query in the URL.

Steps:

- Create a form with a search input field and a submit button.
- When the form is submitted, use the `$_GET` superglobal to retrieve the search query from the URL.
- Display the search query on the results page.

---

3. Simple Contact Form Using `$_REQUEST`

Objective: Create a PHP contact form that accepts both `GET` and `POST` requests using `$_REQUEST`.

Steps:

- Create a form that collects a user's name and message.
- Submit the form using either `GET` or `POST`.
- Use the `$_REQUEST` superglobal to retrieve and display the form data regardless of the method used (GET or POST).

---

4. Display Server Information Using `$_SERVER`

Objective: Display server-related information on a page using the `$_SERVER` superglobal.

Steps:

- Create a PHP script that displays server and environment information such as:
  - The server's IP address.
  - The current script name.
  - The browser (user agent) making the request.
  - The request method used (`GET`, `POST`).
  - The client's IP address.

---

5. Session-based Authentication Using `$_SESSION`

Objective: Create a basic authentication system using `$_SESSION` to store the logged-in user's data.

Steps:

- Create a login form that collects a username and password.
- If the credentials are correct, store the username in a `$_SESSION` variable.
- Create a dashboard page that checks if a user is logged in by checking the session data.
- Create a logout page that destroys the session.

---

6. Cookie-based User Preferences Using `$_COOKIE`

Objective: Use the `$_COOKIE` superglobal to store and retrieve user preferences such as preferred language or theme.

Steps:

- Create a form where users can select their preferred language (e.g., English, Arabic) or theme (e.g., light, dark).
- Store the user's selection in a cookie.
- On page load, retrieve the cookie value using `$_COOKIE` and apply the user's preference to the page (e.g., change the text language or theme).

---

7. File Upload Form Using `$_FILES`

Objective: Create a PHP script that allows users to upload files using the `$_FILES` superglobal.

Steps:

- Create a file upload form where users can select a file from their computer.
- When the form is submitted, use the `$_FILES` superglobal to handle the file upload.
- Display the file's original name, type, and size after the upload.

---

8. Form Security with `$_POST` and CSRF Token

Objective: Create a secure form submission system that prevents **Cross-Site Request Forgery (CSRF)** using a CSRF token stored in `$_SESSION`.

Steps:

- Generate a unique CSRF token and store it in `$_SESSION`.
- Include the CSRF token as a hidden field in a form.
- On form submission, check if the token from the form matches the token stored in the session.
- If the tokens match, process the form. Otherwise, reject the form submission as invalid.

---

9. Redirect with URL Parameters Using `$_GET`

Objective: Create a form that redirects the user to a different page and passes form data using `$_GET` parameters in the URL.

Steps:

- Create a form that collects a user's first name and last name.
- When the form is submitted, redirect the user to a "welcome" page that displays their name using URL parameters (`$_GET`).

---

10. Logging User Activity with `$_SESSION` and `$_SERVER`

Objective: Create a PHP script that logs a user's visit time and IP address using both `$_SESSION` and `$_SERVER`.

Steps:

- When a user visits the page, store the current time in a `$_SESSION` variable.
- Use `$_SERVER` to get the user's IP address.
- Display the visit time and IP address on the page each time the user visits.
- Ensure the visit time is only updated on the first visit by using `$_SESSION`.