1. **Write a script to log the current URL of the page using the window.location property. Extract and log any query parameters. Discuss how query parameters might be exploited in a phishing attack.**

Query parameters can be manipulated in phishing attacks to trick users into revealing sensitive information. Attackers may alter URLs to create fake login pages or redirect users to malicious sites, making it seem like they are interacting with legitimate services. Exploitation of Query Parameters in Phishing Attacks

URL Manipulation: Attackers can modify query parameters in URLs to create deceptive links that appear legitimate. For instance, they might change a URL to include familiar domain names or parameters that mimic trusted services.

Fake Login Pages: By altering query parameters, phishers can direct users to counterfeit login pages that look identical to the real ones. When users enter their credentials, they unknowingly provide sensitive information to the attacker.

Data Harvesting : Malicious URLs can be crafted to collect personal data through forms that seem authentic. Users may be led to believe they are submitting information to a trusted site, while in reality, the data is being sent to the attacker.

Session Hijacking : Attackers can exploit query parameters to hijack user sessions. By manipulating session identifiers in the URL, they can gain unauthorized access to user accounts.

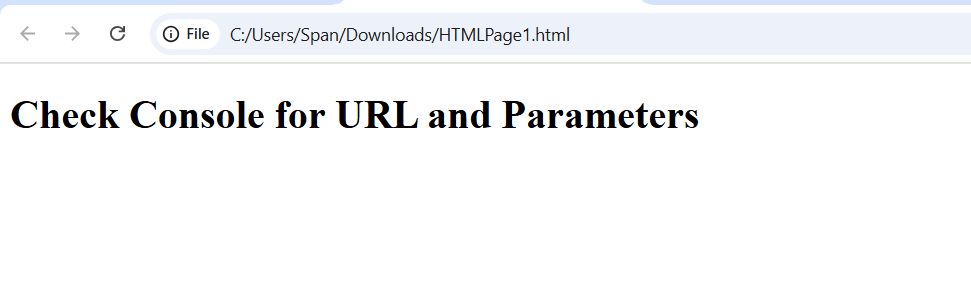
Social Engineering : Phishing attacks often rely on social engineering tactics. By using query parameters that suggest urgency or importance (e.g., "urgent-update" or "account-verification"), attackers can pressure users into acting quickly without verifying the legitimacy of the request.

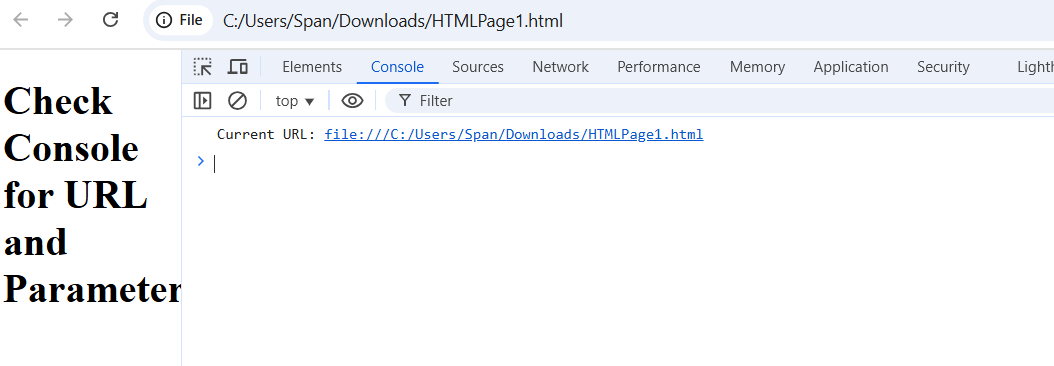
Preventive Measures

User Education : Training users to recognize suspicious URLs and query parameters can significantly reduce the risk of falling victim to phishing attacks.

URL Validation : Implementing strict validation and sanitization of query parameters on the server side can help prevent exploitation.

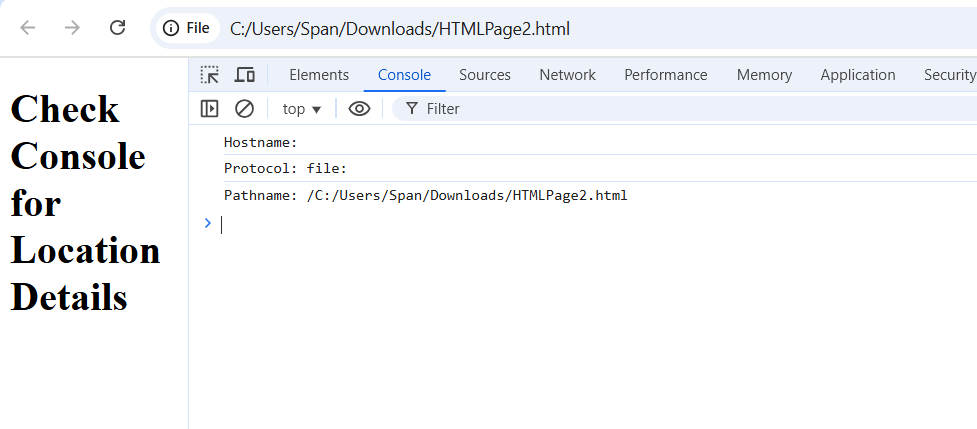
Security Tools : Utilizing security tools that detect and block phishing attempts can provide an additional layer of protection against such attacks.





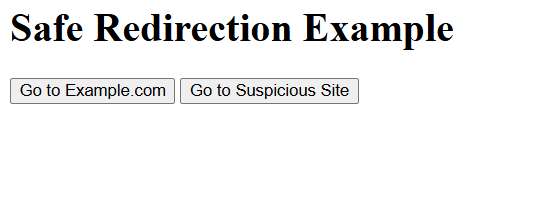
1. **Write a script to log the hostname, protocol, and pathname of the current page using window.location. Explain how attackers might manipulate these properties to perform malicious redirects.**

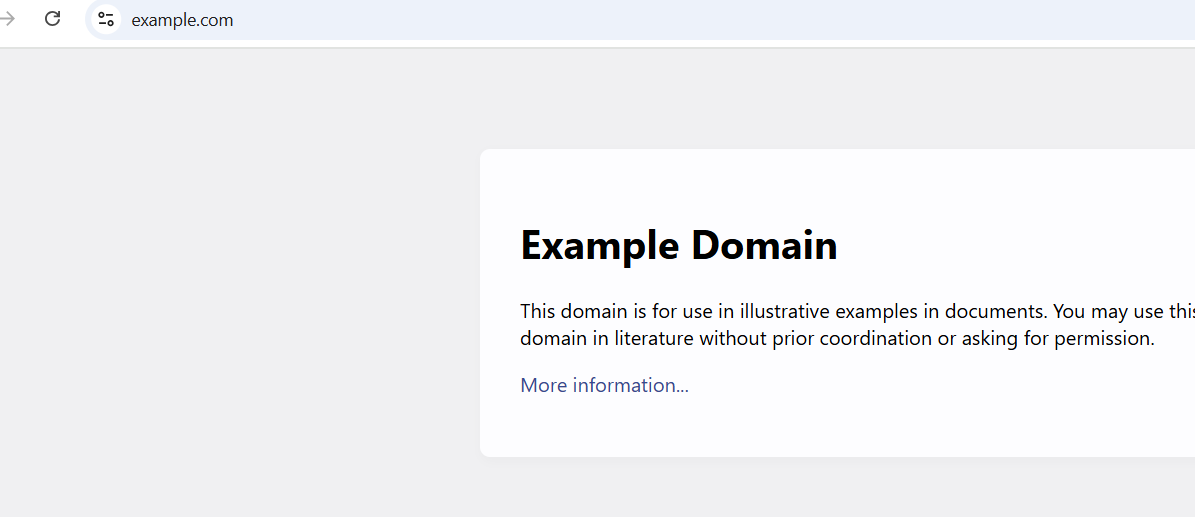
The `window.location` object in JavaScript provides essential information about the current URL, including the hostname, protocol, and pathname. While this functionality is beneficial for web developers, it can also be exploited by attackers to perform malicious redirects. For instance, an attacker might manipulate `window.location` to redirect users to a fraudulent site that mimics a legitimate one, such as a banking login page, thereby tricking users into entering their sensitive information. This manipulation can occur through JavaScript injection, particularly in cases of cross-site scripting (XSS), where an attacker injects malicious code into a web page to alter the location. Additionally, attackers may craft deceptive URLs that appear trustworthy but lead to harmful sites. Therefore, while `window.location` serves a vital role in web development, it poses security risks that necessitate robust protective measures, such as input validation and content security policies, to safeguard users from potential phishing attacks and malware infections.

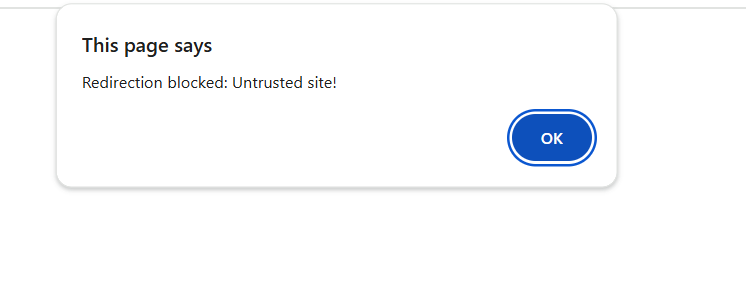


**3.Write a script that uses window.location to redirect users to another webpage. Add a condition to restrict redirects only to trusted domains. Discuss the implications of open redirects in phishing attacks.**

Open redirects are a significant threat in phishing attacks, as they allow attackers to exploit legitimate URLs to mislead users. When a website has an open redirect vulnerability, malicious actors can create links that appear to lead to trusted sites but actually redirect victims to fraudulent pages designed to steal sensitive information, such as login credentials. This not only compromises individual security but also damages the reputation of the legitimate site being exploited, as users may lose trust in its integrity. To combat these risks, organizations must prioritize securing their web applications against open redirects, ensuring that users can navigate the web safely and avoid falling victim to these deceptive schemes.

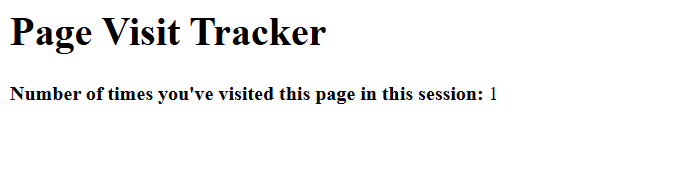


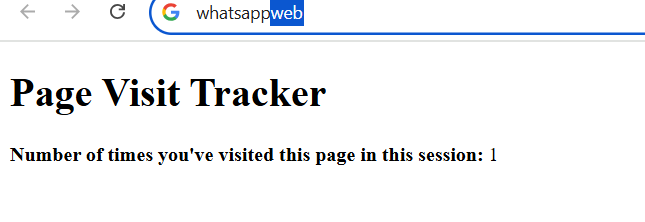


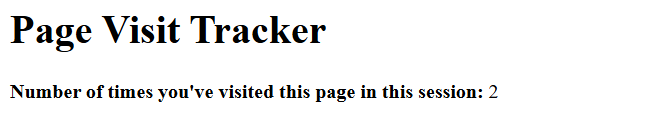


1. **Write a script to log the number of pages visited in the current session using window.history. Discuss potential privacy concerns if malicious scripts attempt to track browser history.**

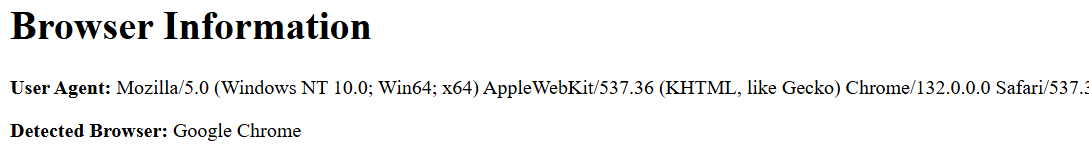
Malicious scripts that attempt to track browser history raise significant privacy concerns for users. Such scripts can collect sensitive information about a user's online behavior, including the websites they visit, the duration of their visits, and their interactions with various content. This data can be exploited for nefarious purposes, such as targeted advertising, identity theft, or even more severe forms of cybercrime. Additionally, users may feel a loss of control over their personal information, leading to a general distrust of online platforms. The potential for misuse of this data underscores the importance of robust security measures and privacy protections to safeguard users from invasive tracking practices and to maintain their trust in digital environments.

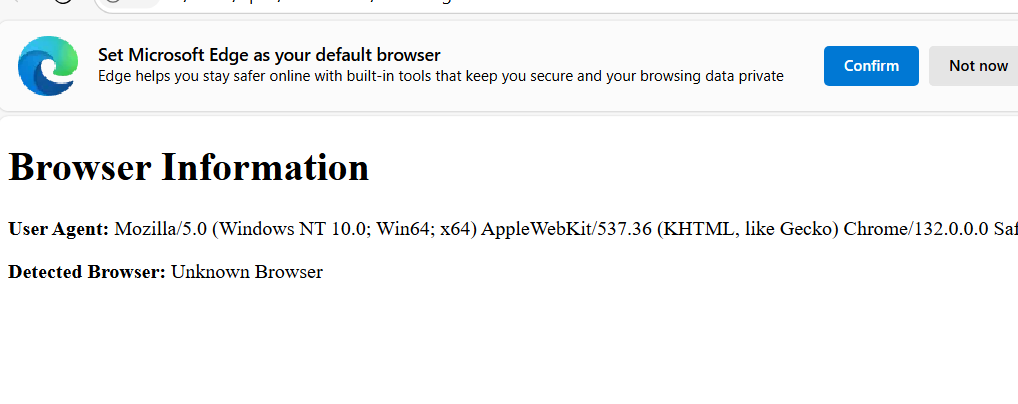




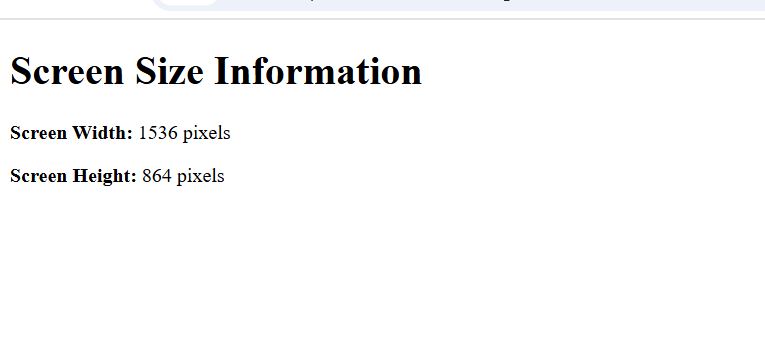
****

**5.Write a script using window.navigator to display the user agent string. Modify the script to identify and log whether the browser is Google Chrome, Mozilla Firefox, or an automation tool.**



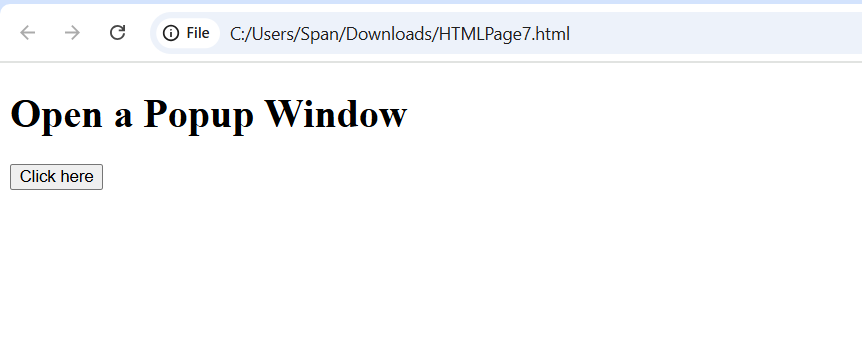
**6. Write a script to log the user's screen width and height using window.screen. Explain how attackers might use screen dimensions to craft phishing attacks.**

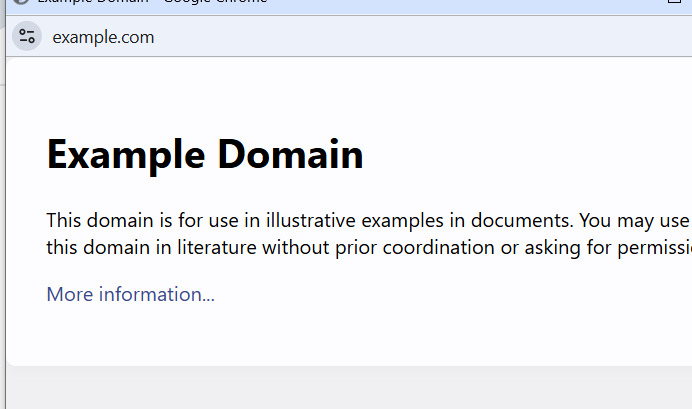
Attackers can exploit screen dimensions in phishing attacks by designing malicious websites that are tailored to specific screen sizes, ensuring they appear legitimate on the victim's device. By manipulating the layout and elements based on the user's screen resolution, attackers can create a more convincing interface that mimics trusted sites, increasing the likelihood of user interaction and data theft. This tactic can involve adjusting the size of pop-up windows or forms to fit seamlessly within the user's display, making it harder for victims to recognize that they are not interacting with a genuine site. Additionally, attackers may use responsive design techniques to ensure that their phishing pages look professional and trustworthy, regardless of the device being used. By leveraging these visual cues, they can effectively lower the user's guard, leading to higher success rates in capturing sensitive information like login credentials or personal data.



**7.Write a script to open a new popup window using window.open. Discuss the risks associated with popup-based attacks and suggest mitigation strategies.**

Popup-based attacks pose significant risks, including the potential for users to inadvertently download malware, disclose sensitive information, or fall victim to phishing schemes. These attacks often exploit the trust users place in familiar interfaces, making it crucial to implement mitigation strategies such as using robust popup blockers, educating users about recognizing suspicious popups, and ensuring that security software is up-to-date to detect and block malicious content. Additionally, organizations can enhance their defenses by regularly updating their web browsers and plugins, as outdated software can be more susceptible to exploitation. Implementing strict access controls and monitoring user behavior can also help identify unusual activities that may indicate a popup-based attack. Encouraging users to verify the legitimacy of any popup before interacting with it, especially those requesting personal information or prompting downloads, can further reduce the risk of falling prey to these deceptive tactics. By fostering a culture of security awareness and vigilance, organizations can significantly mitigate the threats posed by popup-based attacks.

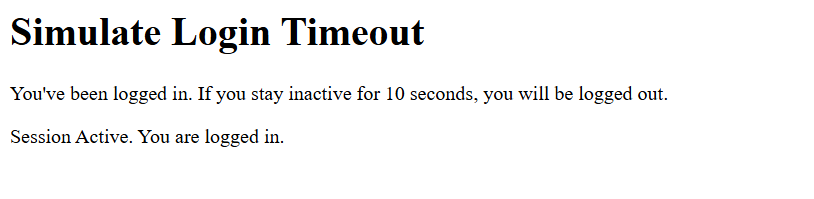


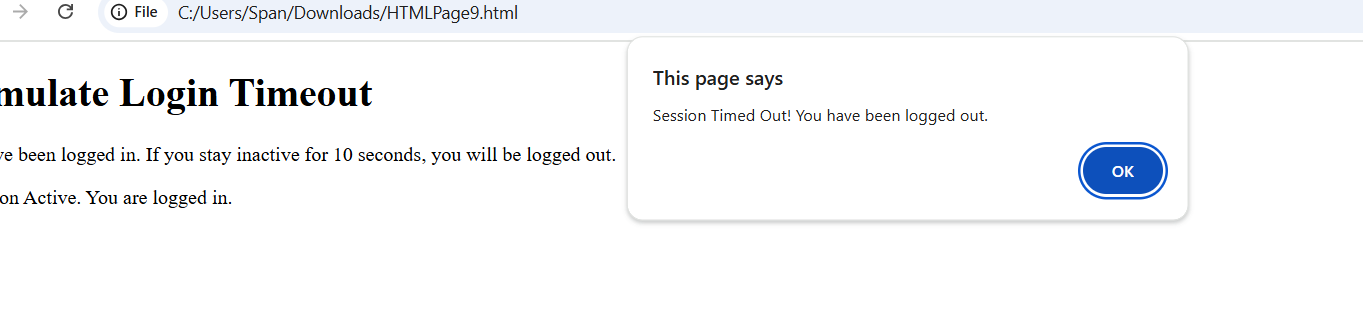


**8.Write a script to log the domain and origin of the current page using window.location. Explain how these properties are relevant in CORS security.**

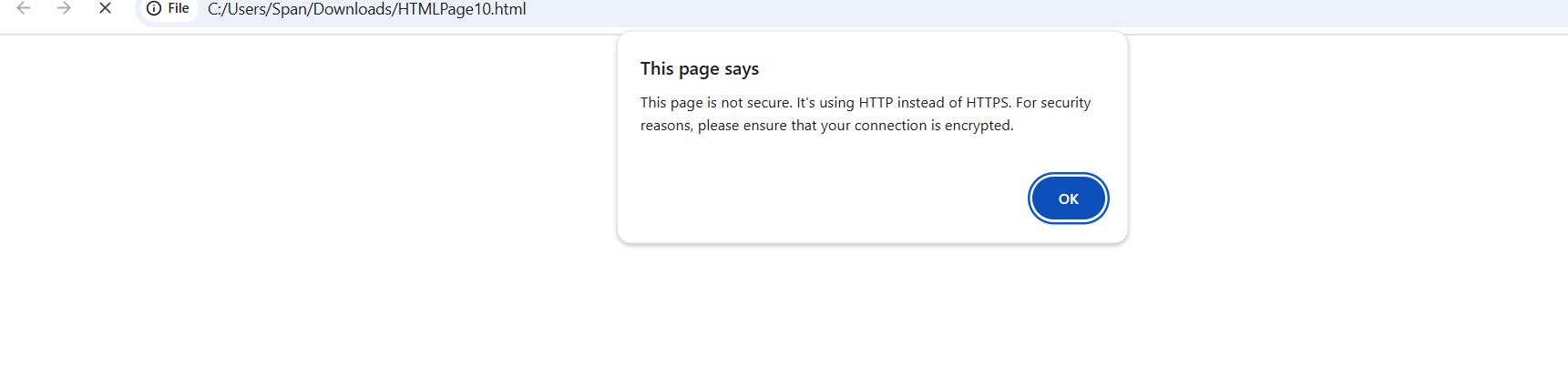
Cross-Origin Resource Sharing (CORS) is a security feature that allows web applications to request resources from different origins while maintaining a level of protection against malicious activities. The properties of the `window.location` object—such as hostname, protocol, and pathname—are crucial in this context because they help determine the origin of a request. When a web application makes a cross-origin request, the browser checks the origin of the request against the server's CORS policy. If the origin matches the allowed list specified by the server, the request is permitted; otherwise, it is blocked. This mechanism helps prevent unauthorized access to sensitive data and protects users from cross-site attacks. By carefully managing these properties and implementing strict CORS policies, developers can enhance the security of their applications and safeguard user data from potential threats.

**9.Write a script using window.setTimeout that simulates a login timeout after 10 seconds of inactivity. Enhance the script to reset the timer whenever the user interacts with the page.**





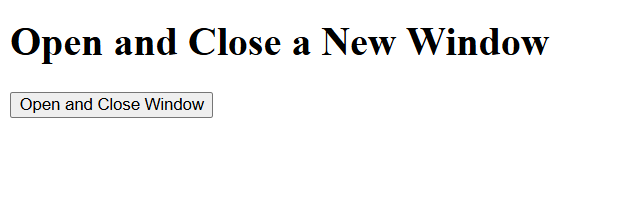
**10.Write a script that checks if the current URL uses "https" using window.location. Alert the user if the page is not secure and suggest measures to handle such scenarios.**

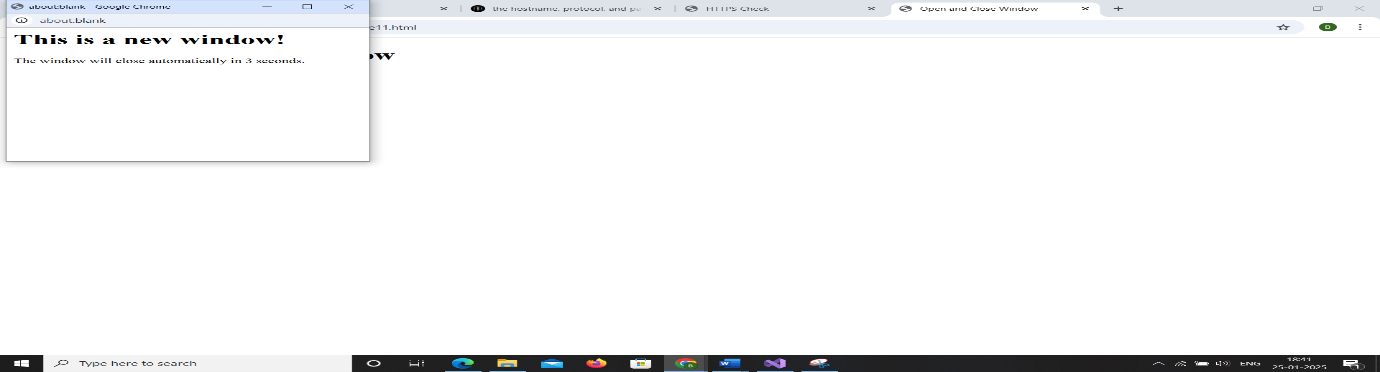


**11.Write a script to open a new window and then close it using window.close.**

**Discuss scenarios where malicious scripts might misuse this functionality to disrupt user experience**.

Malicious scripts can significantly disrupt user experience by exploiting functionalities like popups, redirects, and even browser APIs. For instance, an attacker might deploy a script that generates endless popups, overwhelming the user and making it nearly impossible to navigate away from the page. This tactic can lead to frustration and confusion, potentially causing users to inadvertently disclose personal information in a panic. Additionally, scripts can hijack browser controls, redirecting users to unwanted sites or displaying misleading content that mimics legitimate services. Such disruptions not only compromise the user’s online safety but also erode trust in the website, making it essential for developers to implement robust security measures to protect users from these malicious tactics.



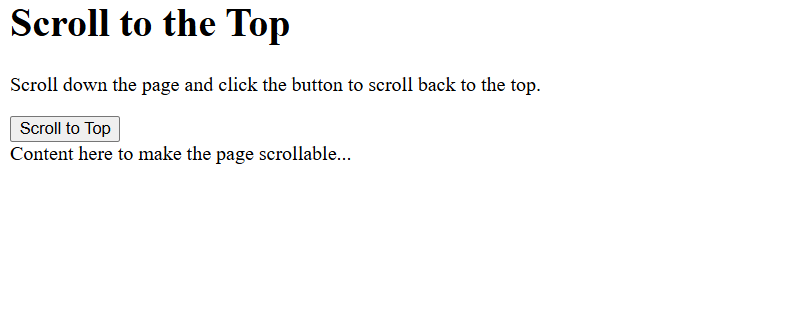


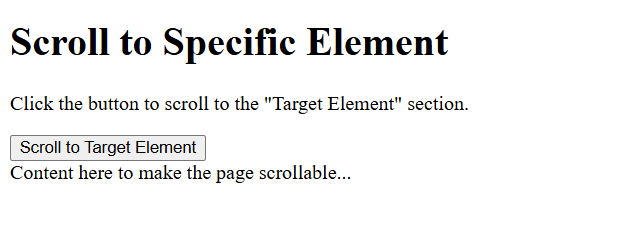
**12.Write a script to scroll the webpage to the top using window.scrollTo.**

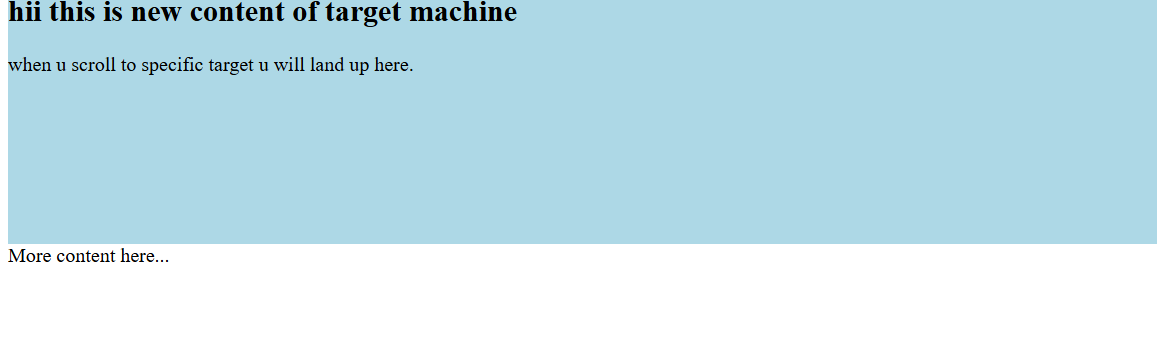
**Modify the script to scroll to a specific element on the page.**

**Discuss how attackers might use scrolling to hide malicious content or overlays.**

Attackers can cleverly exploit scrolling techniques to hide malicious content or overlays, making it difficult for users to notice the deception. For example, they might create a webpage where harmful elements, such as phishing forms or misleading advertisements, are positioned off-screen or behind legitimate content. As users scroll down the page, they may inadvertently reveal these hidden overlays, which can prompt them to enter sensitive information or click on dangerous links without realizing the risk. This tactic takes advantage of user behavior and can lead to significant security breaches, as victims may feel they are interacting with a trustworthy site while being manipulated into compromising their personal data. To combat this, web developers must prioritize transparency and user awareness, ensuring that all content is clearly visible and that users are educated about potential threats lurking in seemingly innocuous scrolling experiences.







**13. Write a script to retrieve the current webpage's title using document.title and log it to the console. Modify the title to include a security warning if it doesn't already contain "Secure." Explain how attackers might manipulate the document title for phishing or social engineering attacks.**

Attackers often manipulate document titles to create a false sense of legitimacy, making phishing attempts more convincing. By using familiar or official-sounding titles, they can trick users into believing the document is safe or important, increasing the likelihood that victims will open it and provide sensitive information or click on malicious links. This tactic can involve using titles that mimic legitimate documents, such as "Invoice," "Payment Confirmation," or "Urgent Security Update," which can evoke a sense of urgency or importance. When users see these titles, they may not scrutinize the source or content closely, leading them to act impulsively. This manipulation plays on human psychology, leveraging trust and familiarity to bypass critical thinking, ultimately resulting in successful social engineering attacks that compromise sensitive data.