Day 3

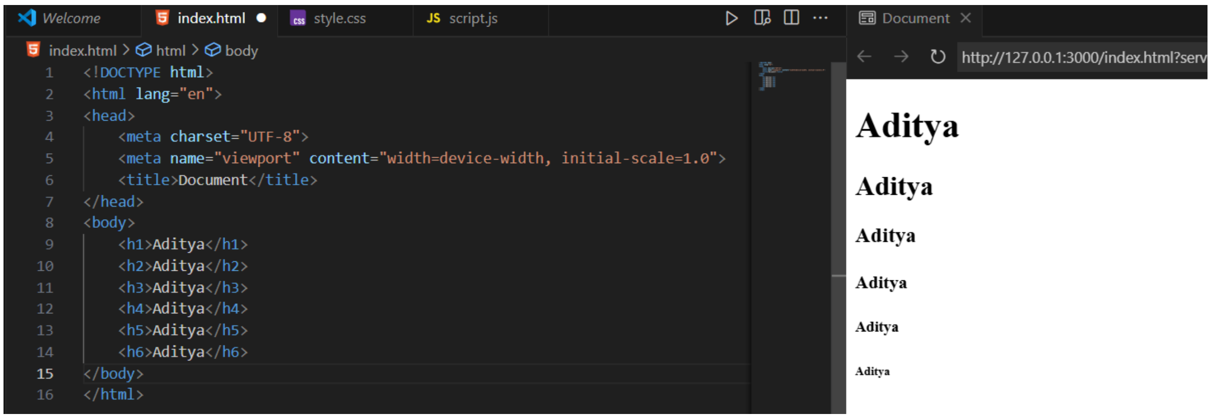
"Web Development + Security"

Headings:

What are Headings in HTML?

In HTML, a heading is a tag used to define titles or subtitles on a webpage. They range from <h1> (the most important, main heading) to <h6> (the least important, smallest heading).

Example:



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Aditya</h1>
    <h2>Aditya</h2>
    <h3>Aditya</h3>
    <h4>Aditya</h4>
    <h5>Aditya</h5>
    <h6>Aditya</h6>
</body>
</html>
```

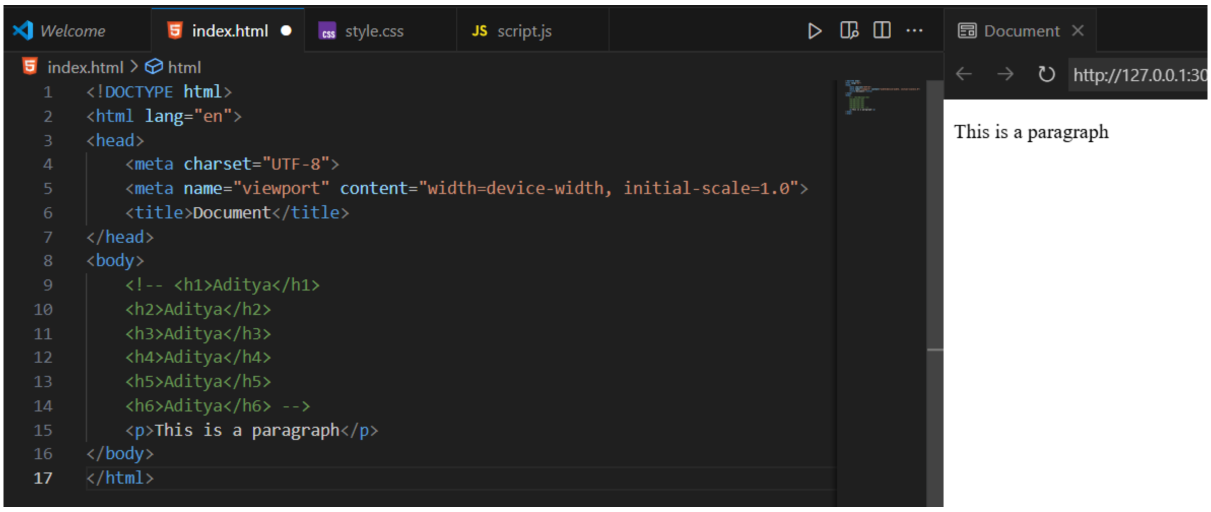
```
Aditya
Aditya
Aditya
Aditya
Aditya
Aditya
```

Paragraphs:

What are Paragraphs in HTML?

In HTML, a paragraph is defined using the <p> tag. It is used to display blocks of text as separate sections, making the content easier to read. Browsers automatically add some space (margin) before and after each paragraph.

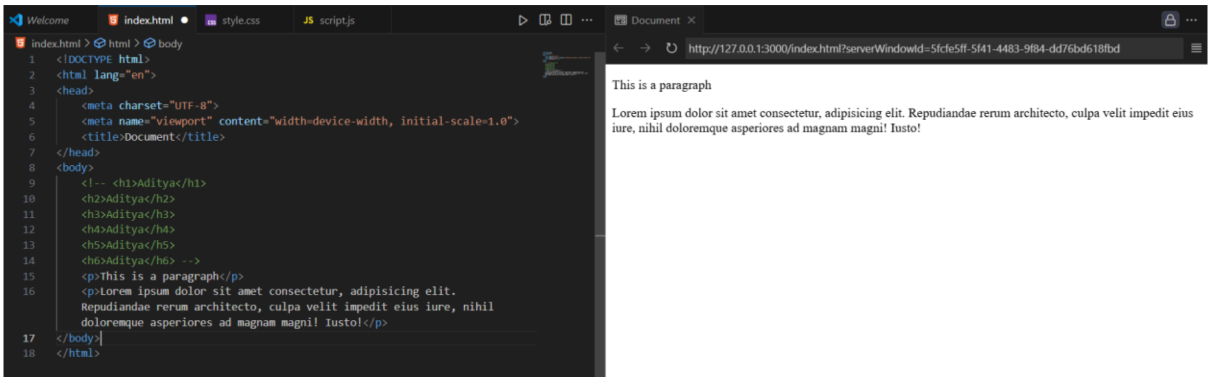
Example:



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <!-- <h1>Aditya</h1>
    <h2>Aditya</h2>
    <h3>Aditya</h3>
    <h4>Aditya</h4>
    <h5>Aditya</h5>
    <h6>Aditya</h6> -->
    <p>This is a paragraph</p>
</body>
</html>
```

```
This is a paragraph
```

We can also use Lorem keyword to write some text, example:



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <!-- <h1>Aditya</h1>
    <h2>Aditya</h2>
    <h3>Aditya</h3>
    <h4>Aditya</h4>
    <h5>Aditya</h5>
    <h6>Aditya</h6> -->
    <p>This is a paragraph</p>
    <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit.
    Repudiandae rerum architecto, culpa velit impedit eius iure, nihil
    doloremque asperiores ad magnam magni! Iusto!</p>
</body>
</html>
```

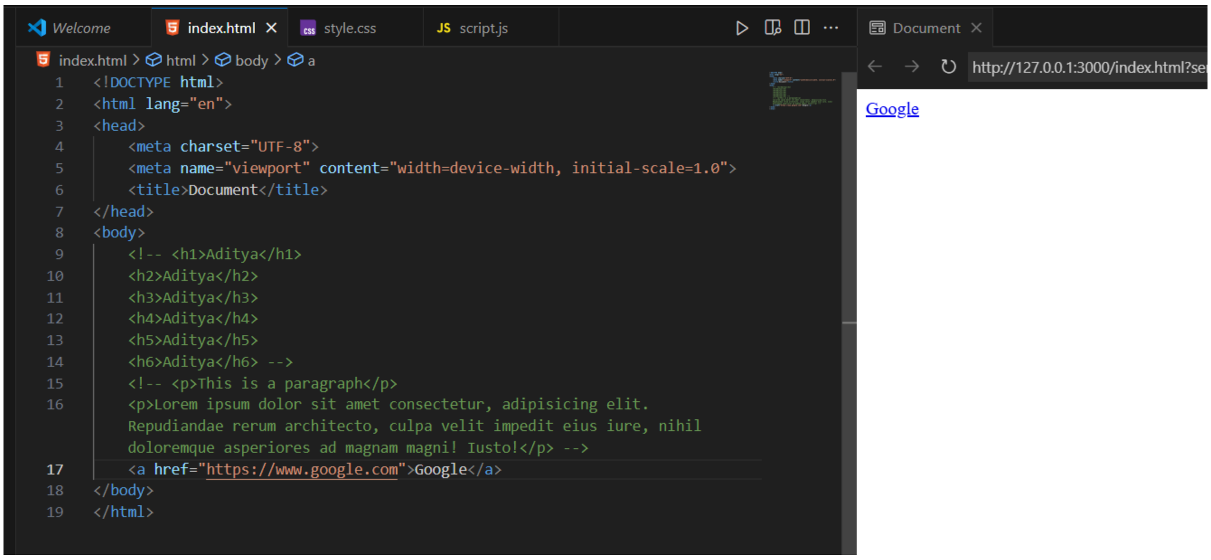
```
http://127.0.0.1:3000/index.html?serverWindowId=5fcfe5ff-5f41-4483-9f84-dd76bd618fbd

This is a paragraph

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Repudiandae rerum architecto, culpa velit impedit eius
iure, nihil doloremque asperiores ad magnam magni! Iusto!
```

Links:

What are links in HTML?

In HTML, links are created using the <a> tag (called an anchor tag). They let you navigate from one page to another, open files, or jump to sections on the same page.

Example:

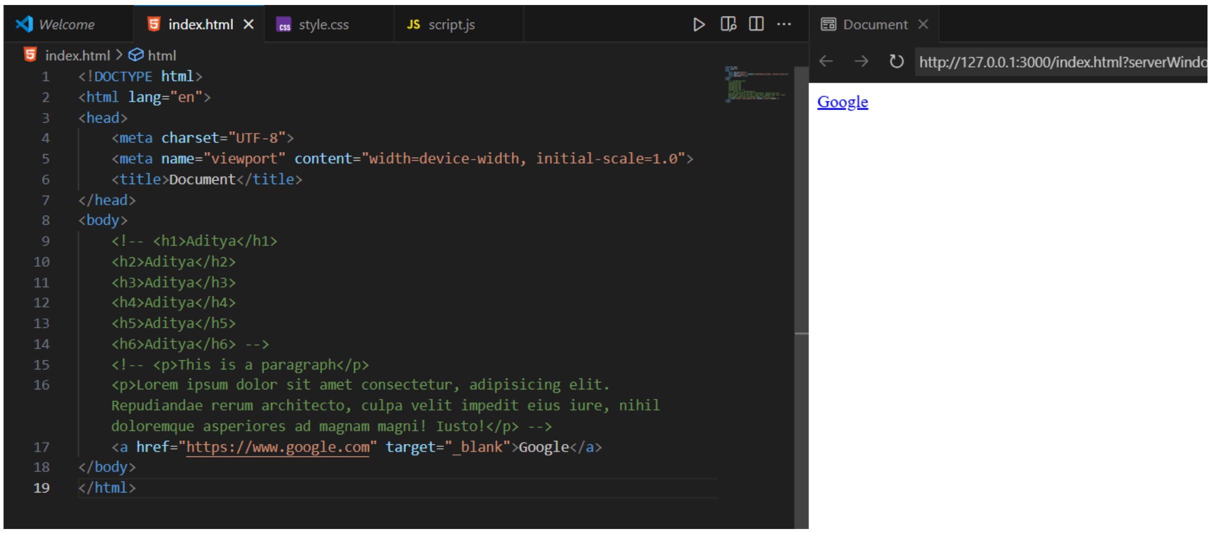


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <!-- <h1>Aditya</h1>
    <h2>Aditya</h2>
    <h3>Aditya</h3>
    <h4>Aditya</h4>
    <h5>Aditya</h5>
    <h6>Aditya</h6> -->
    <!-- <p>This is a paragraph</p>
    <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit.
    Repudiandae rerum architecto, culpa velit impedit eius iure, nihil
    doloremque asperiores ad magnam magni! Iusto!</p> -->
    <a href="https://www.google.com">Google</a>
</body>
</html>
```

```
http://127.0.0.1:3000/index.html?se

Google
```

In above example if we click on the link, it will open the google.com in the same page. In case we wish that it should open in the new tab, we can use the attribute of <a> tag called ‘target’.

Example:



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <!-- <h1>Aditya</h1>
    <h2>Aditya</h2>
    <h3>Aditya</h3>
    <h4>Aditya</h4>
    <h5>Aditya</h5>
    <h6>Aditya</h6> -->
    <!-- <p>This is a paragraph</p>
    <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit.
    Repudiandae rerum architecto, culpa velit impedit eius iure, nihil
    doloremque asperiores ad magnam magni! Iusto!</p> -->
    <a href="https://www.google.com" target="_blank">Google</a>
</body>
</html>
```

Google

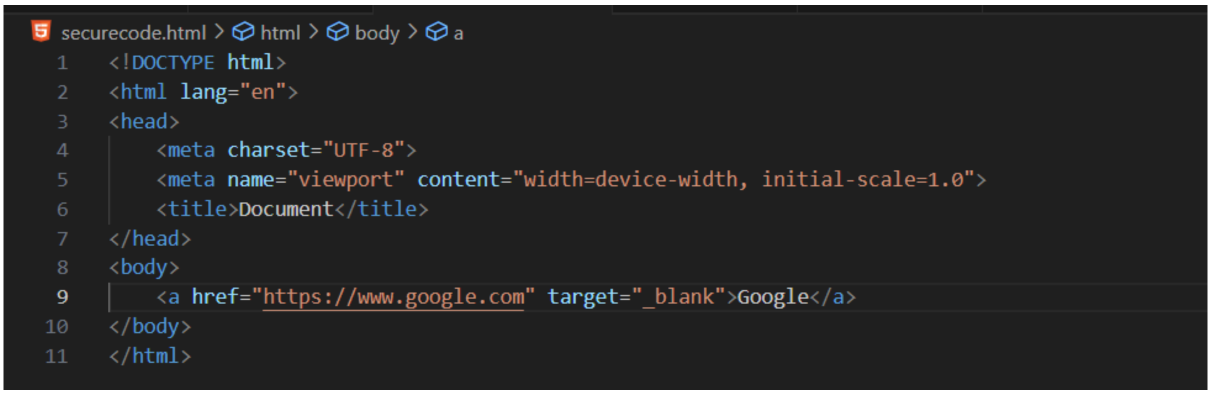
Important attributes of <a> tag:

Attribute	**Use Case**	**Example**
href	Defines the link destination (URL or file).	<a href="https://example.com">Visit Example</a>
target	Controls where the link opens. Common: _self (same tab), _blank (new tab).	<a href="page.html" target="_blank">Open in new tab</a>
rel	Adds relationship info; improves **security & SEO**. Common: noopener, noreferrer, nofollow.	<a href="https://example.com" target="_blank" rel="noopener noreferrer">Secure Link</a>
title	Shows a tooltip when hovering over the link.	<a href="about.html" title="More about us">About Us</a>
download	Makes the browser download the file instead of opening it.	<a href="file.pdf" download>Download PDF</a>
href="#"	Placeholder link (used with JavaScript or empty link).	<a href="#">Click Me</a>

Secure Practices for <a>:

1. **Always use HTTPS links**
   - Prevents man-in-the-middle attacks.

Example:



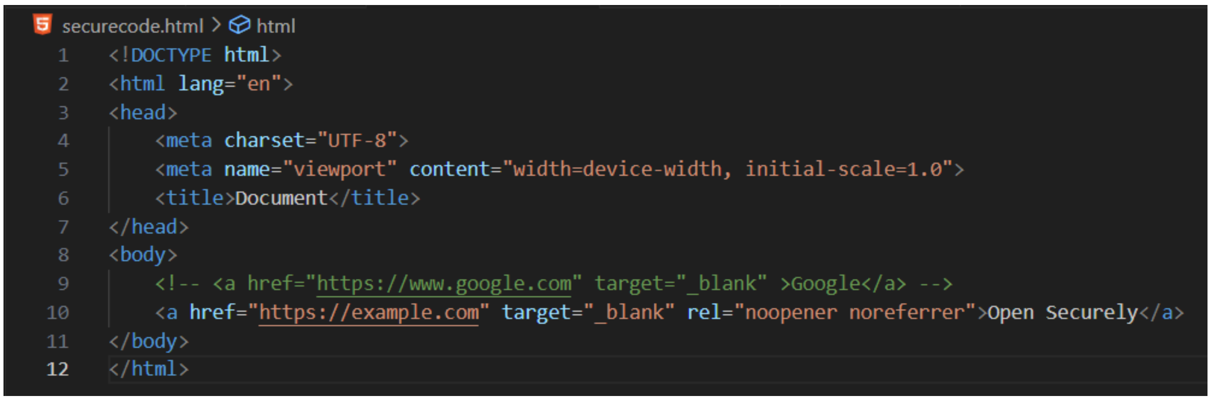
securecode.html > html > body > a

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <a href="https://www.google.com" target="_blank">Google</a>
</body>
</html>
```

2. **When using target="_blank", add rel="noopener noreferrer"**

When you open a link in a new tab using target="_blank", the new page can access the original page with window.opener, which attackers can exploit to redirect you to a fake site (called reverse tabnabbing). Adding rel="noopener noreferrer" breaks that connection, making the new tab independent and keeping your original page safe.

Example:



securecode.html > html

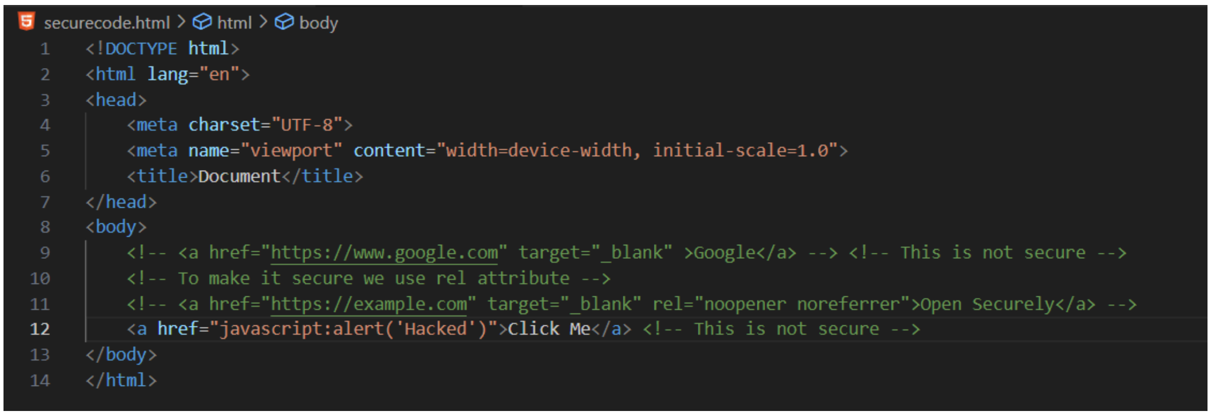
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <!-- <a href="https://www.google.com" target="_blank" >Google</a> -->
    <a href="https://example.com" target="_blank" rel="noopener noreferrer">Open Securely</a>
</body>
</html>
```

3. Validate or sanitize dynamic links

If a link comes from user input, attackers could insert javascript: or data: schemes to run malicious code. Always check that links start with trusted protocols like https://.

Example:

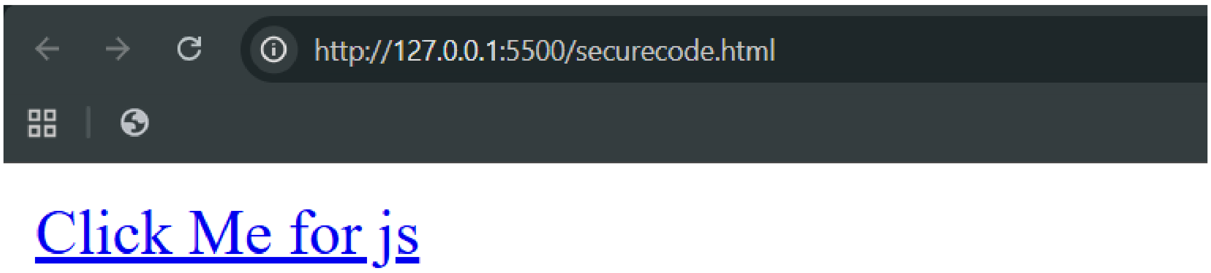
Unsafe:



securecode.html > html > body

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <!-- <a href="https://www.google.com" target="_blank" >Google</a> --> <!-- This is not secure -->
    <!-- To make it secure we use rel attribute -->
    <!-- <a href="https://example.com" target="_blank" rel="noopener noreferrer">Open Securely</a> -->
    <a href="javascript:alert('Hacked')">Click Me</a> <!-- This is not secure -->
</body>
</html>
```

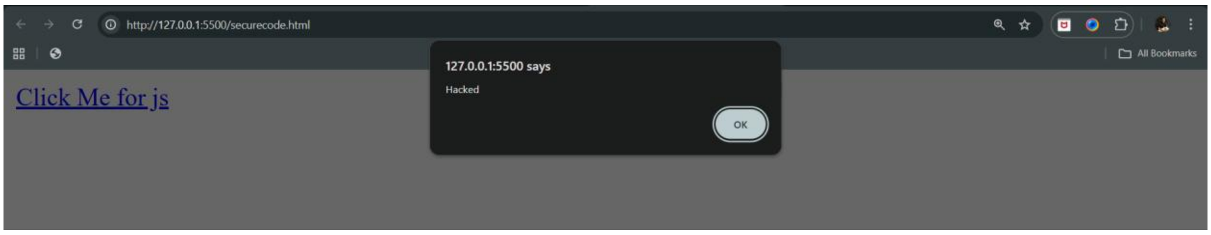
Output:



http://127.0.0.1:5500/securecode.html

Click Me for js

When clicked:



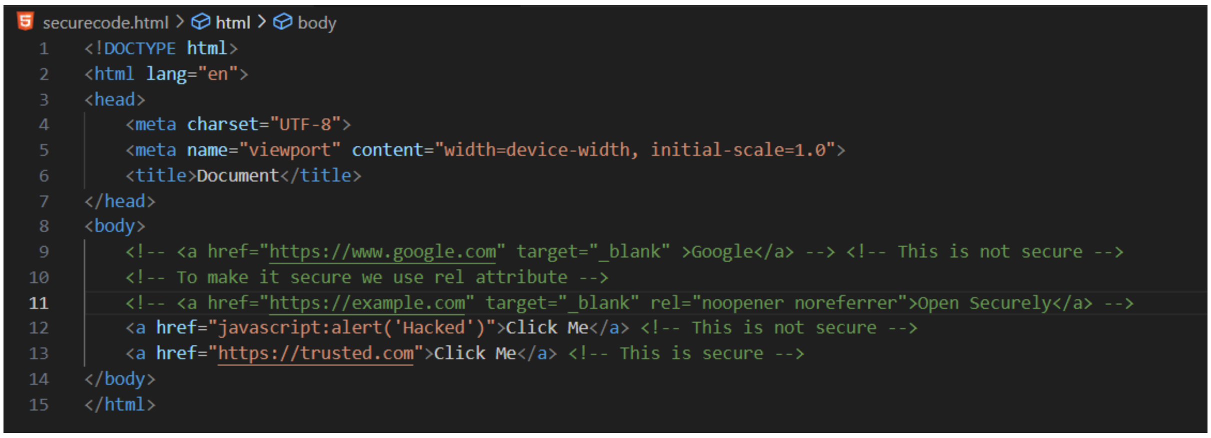
http://127.0.0.1:5500/securecode.html

Click Me for js

127.0.0.1:5500 says

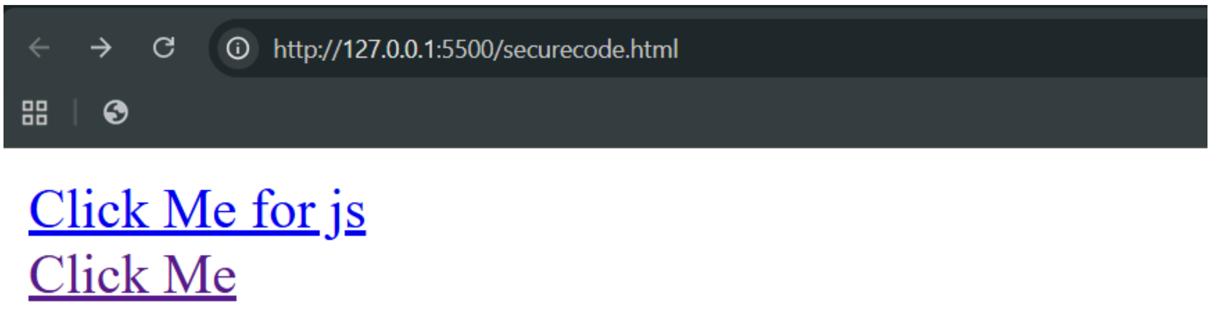
Hacked

Safe:



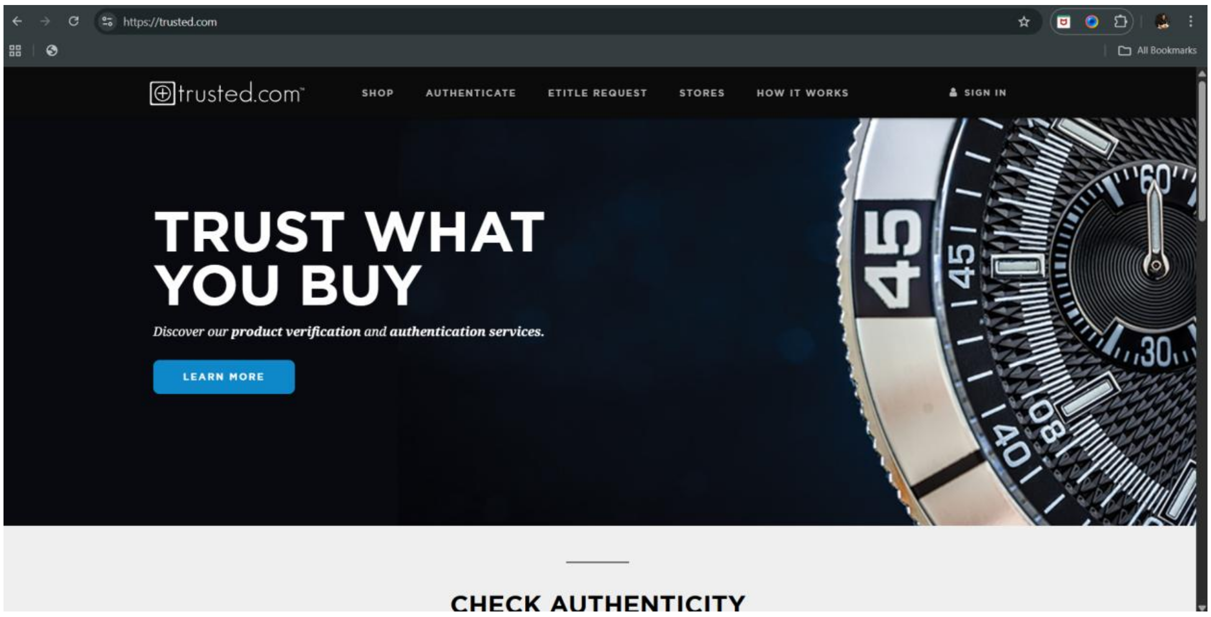
```
securecode.html > html > body
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <!-- <a href="https://www.google.com" target="_blank" >Google</a> --> <!-- This is not secure -->
    <!-- To make it secure we use rel attribute -->
    <!-- <a href="https://example.com" target="_blank" rel="noopener noreferrer">Open Securely</a> -->
    <a href="javascript:alert('Hacked')">Click Me</a> <!-- This is not secure -->
    <a href="https://trusted.com">Click Me</a> <!-- This is secure -->
</body>
</html>
```

Output:



```
http://127.0.0.1:5500/securecode.html

Click Me for js
Click Me
```

When clicked on 2nd link:



```
https://trusted.com
```

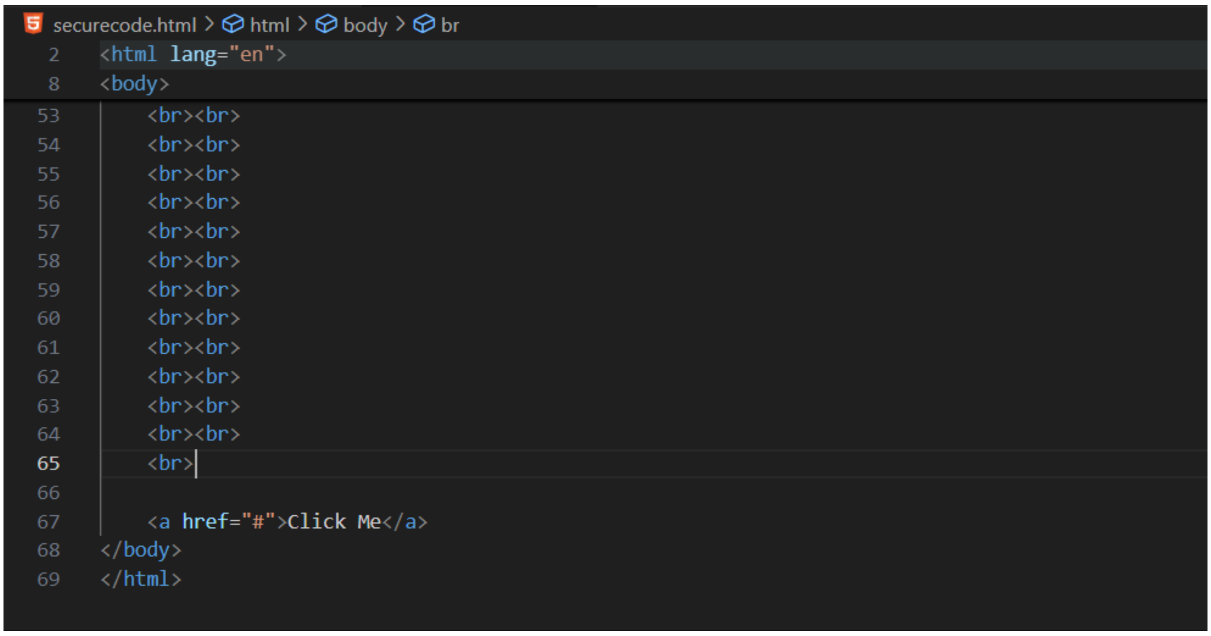
4. Avoid placeholder href="#" without JavaScript handling

Using href="#" without handling causes the page to jump to the top unnecessarily. If it's just for an action (like a button), use a <button> instead.

When you use href="#" in a link, clicking it will make the page jump back to the very top, which can be annoying if you didn't want that. Developers sometimes use it as a placeholder, but if your link is not meant to take users to another page and is just for an action (like submit, toggle, or click events), it's better to use a <button> instead. Buttons don't cause the page to jump, and they are the correct element for actions.

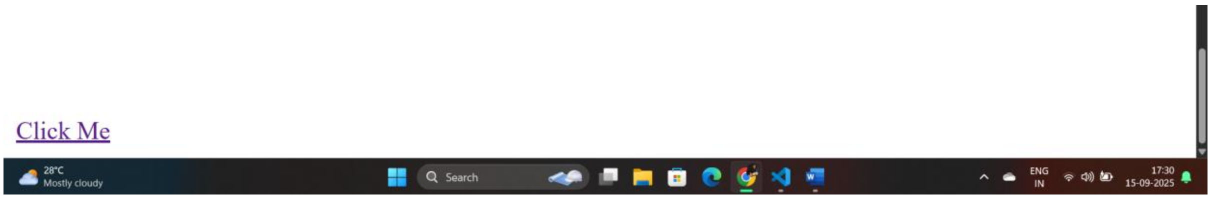
Example:

Bad practice:

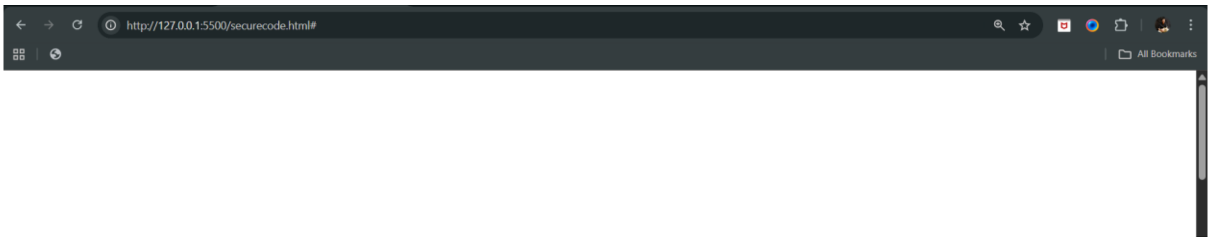


```
securecode.html > html > body > br
<html lang="en">
<body>
        <br><br>
        <br><br>
        <br><br>
        <br><br>
        <br><br>
        <br><br>
        <br><br>
        <br><br>
        <br><br>
        <br><br>
        <br><br>
        <br><br>
        <br>

        <a href="#">Click Me</a>
</body>
</html>
```

Output: We are at bottom of screen due to many <br> tags.



When clicked: we reached top of the page.

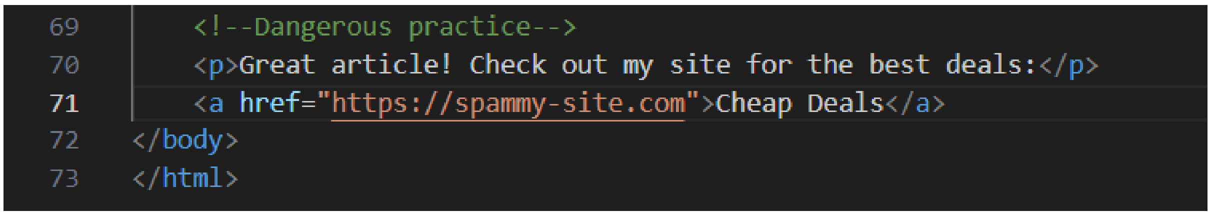


5. **Use rel="nofollow" for untrusted/external links**

If you run a blog with comments, spammers might add links like <a href="https://spammy-site.com">Cheap Deals</a>. Without protection, search engines think you endorse that site, which can hurt your SEO and credibility. By adding rel="nofollow"—<a href="https://spammy-site.com" rel="nofollow">Cheap Deals</a>—Google won't pass ranking power to the spammy site, keeping your blog's reputation safe while still letting users click if they choose.

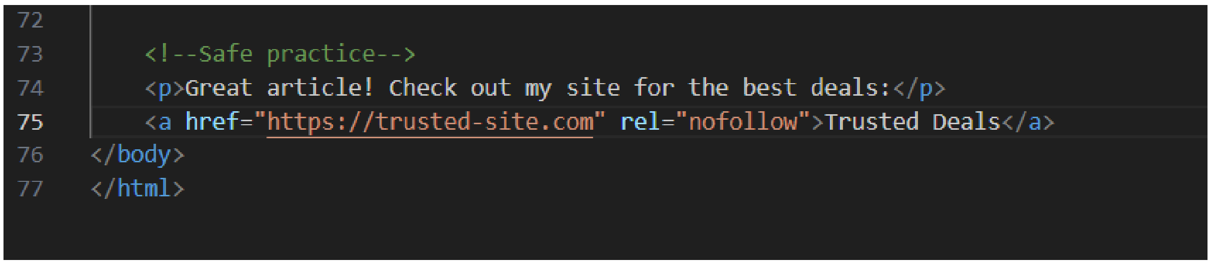
Example:

Bad practice:



```html
    <!--Dangerous practice-->
    <p>Great article! Check out my site for the best deals:</p>
    <a href="https://spammy-site.com">Cheap Deals</a>
</body>
</html>
```

Good practice:

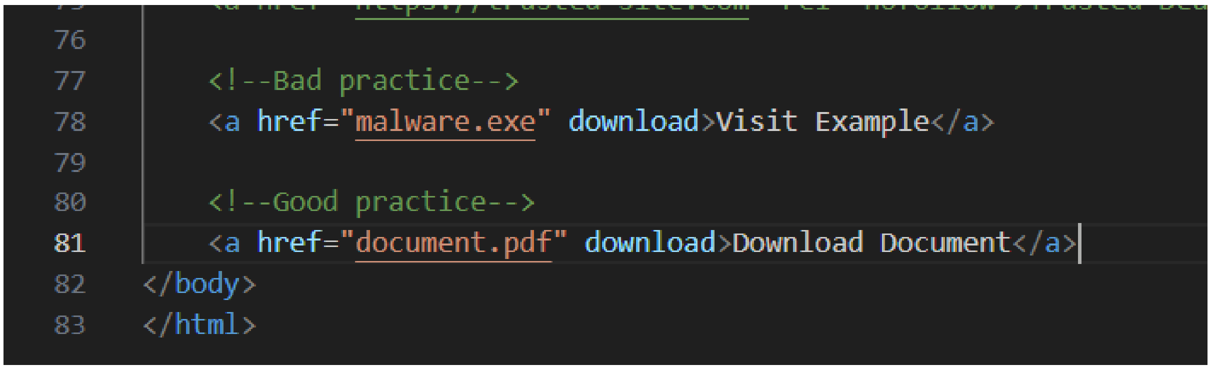


```html

    <!--Safe practice-->
    <p>Great article! Check out my site for the best deals:</p>
    <a href="https://trusted-site.com" rel="nofollow">Trusted Deals</a>
</body>
</html>
```

6. **Be careful with file downloads (download attribute)**

The download attribute forces file download, but attackers could trick users with harmful files. Only allow trusted files.

Example:



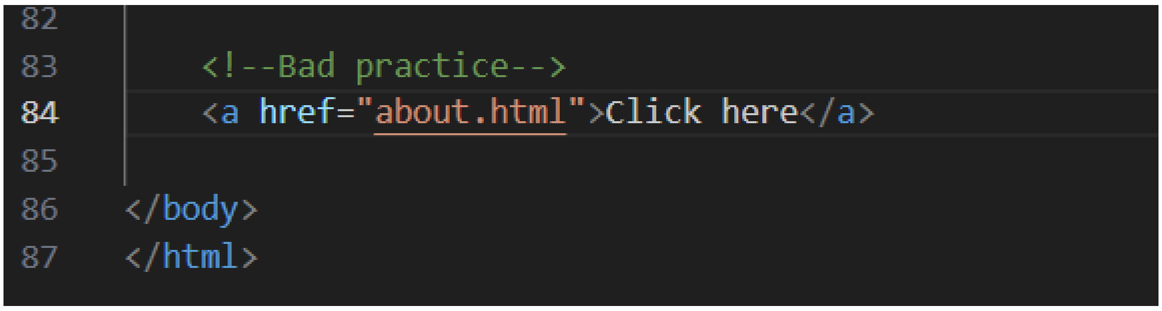
```html

    <!--Bad practice-->
    <a href="malware.exe" download>Visit Example</a>

    <!--Good practice-->
    <a href="document.pdf" download>Download Document</a>
</body>
</html>
```

7. Add descriptive title or aria-label

The title or aria-label attribute makes links clearer for users and screen readers, preventing confusion or phishing tricks.

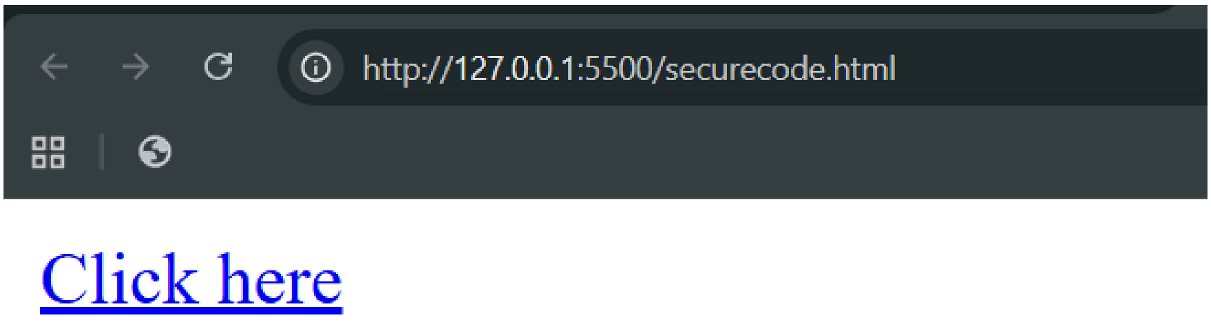
Example:

Bad practice:



```
    <!--Bad practice-->
    <a href="about.html">Click here</a>

</body>
</html>
```

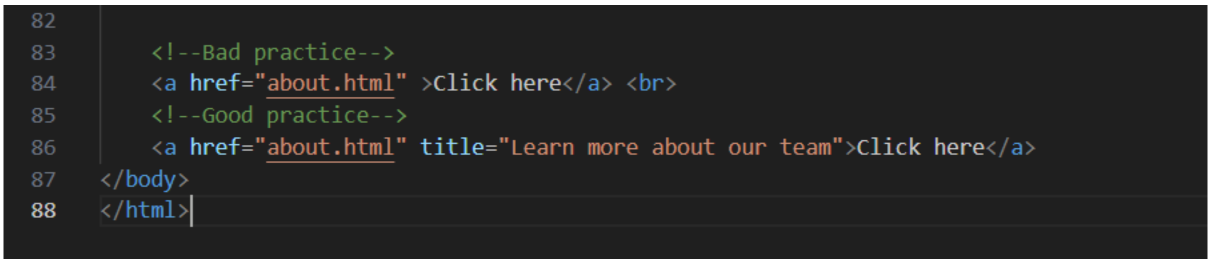
Output: Nothing shows on hover



http://127.0.0.1:5500/securecode.html

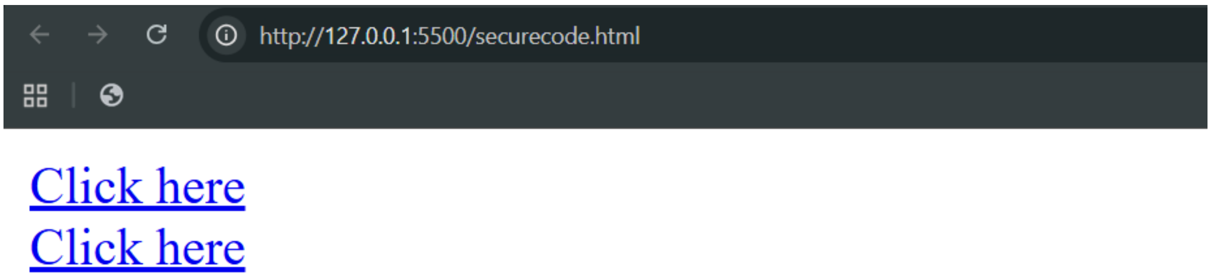
Click here

Good practice:



```
    <!--Bad practice-->
    <a href="about.html" >Click here</a> <br>
    <!--Good practice-->
    <a href="about.html" title="Learn more about our team">Click here</a>
</body>
</html>
```

Output: When we hover on the 2nd link it will show the text “Learn more about our team”



Click here
Click here

8. Restrict referrer info with rel="noreferrer"

When linking to another site, browsers normally send your page’s URL as a referrer. Use rel="noreferrer" to hide it if the URL has sensitive data.

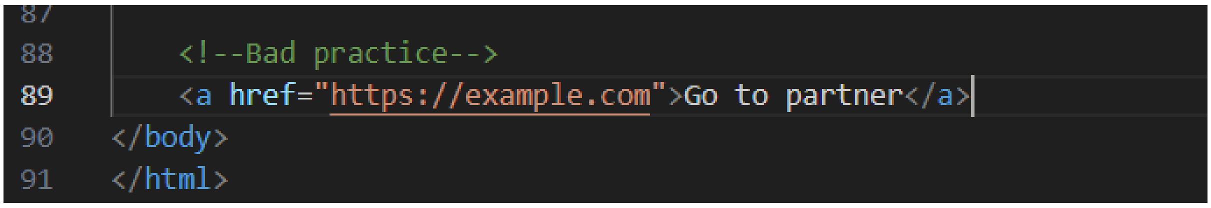
Example:

Bad Practice:

By default, when a user clicks a link, the browser sends the current page’s URL as the referrer to the new site. If your page URL is:

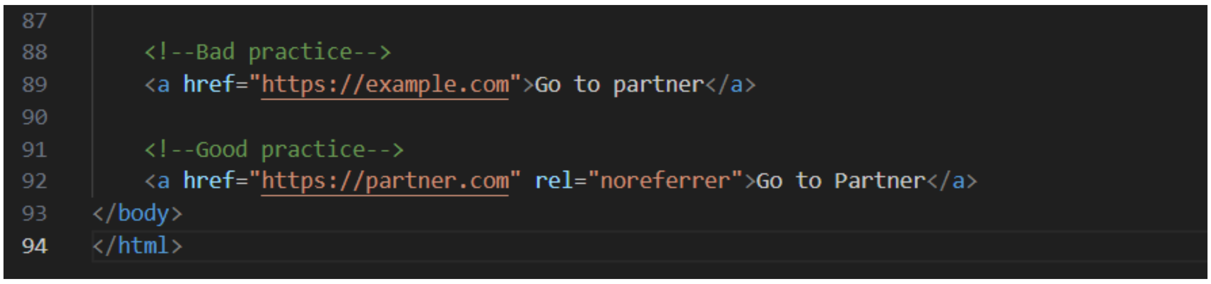
https://myshop.com/checkout?user=john&card=1234

then example.com will receive that full referrer URL, including sensitive info (user, card).



```html
    <!--Bad practice-->
    <a href="https://example.com">Go to partner</a>
</body>
</html>
```

Good practice:



```html
    <!--Bad practice-->
    <a href="https://example.com">Go to partner</a>

    <!--Good practice-->
    <a href="https://partner.com" rel="noreferrer">Go to Partner</a>
</body>
</html>
```

--The End--