WS 3.1 - Client-Side Security

Polonium - Pwnzer0tt1

`gh repo fork WS_3.1 - Client-Side Security`

Prerequisites

- WS_1.1 - HTTP Protocol and Web-Security Overview
- Knowledge about:
  - HTML
  - JavaScript
  - Esperanto

Outline

- XSS
- CSRF

Overview - HTML

Source: https://developer.mozilla.org/en-US/docs/Web/HTML

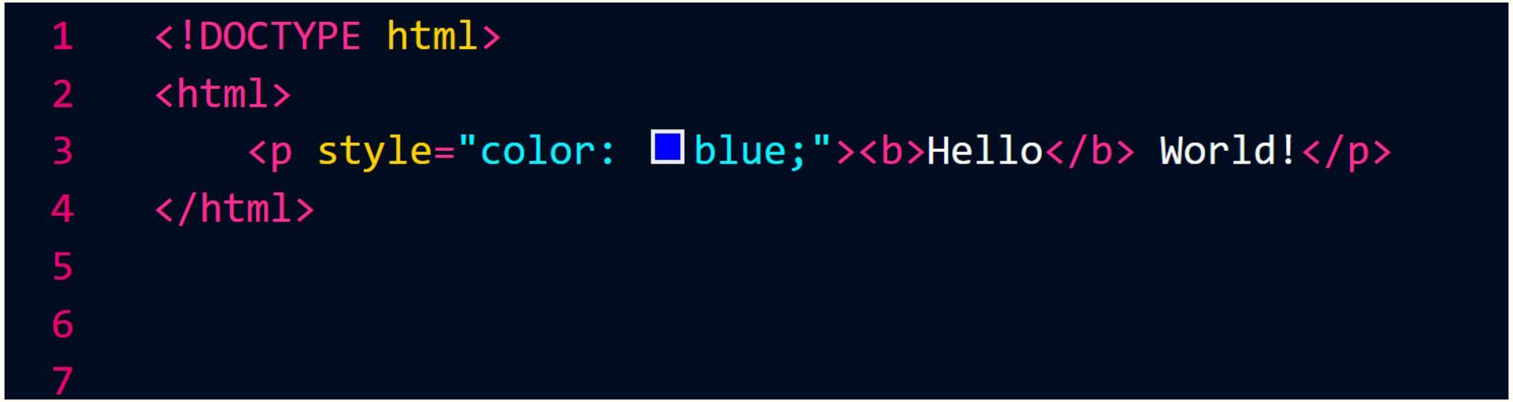
HyperText Markup Language (HTML) is the language used to create web pages, it defines the meaning and structure of web content.

Hypertext refers to links that connect web pages to one another. Links are a fundamental aspect of the Web.

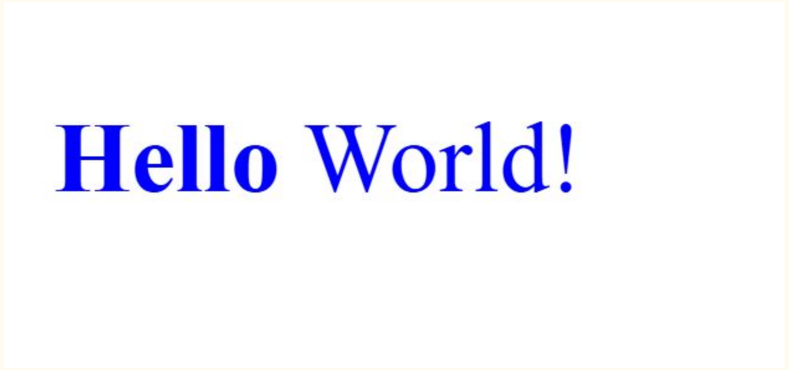
HTML uses **markup** to annotate text, image and other content for display in a Web browser.

HTML includes special elements defined by **tags** which consist of the element name surrounded by <>.

Overview - HTML



```
<!DOCTYPE html>
<html>
    <p style="color: ■blue;"><b>Hello</b> World!</p>
</html>
```



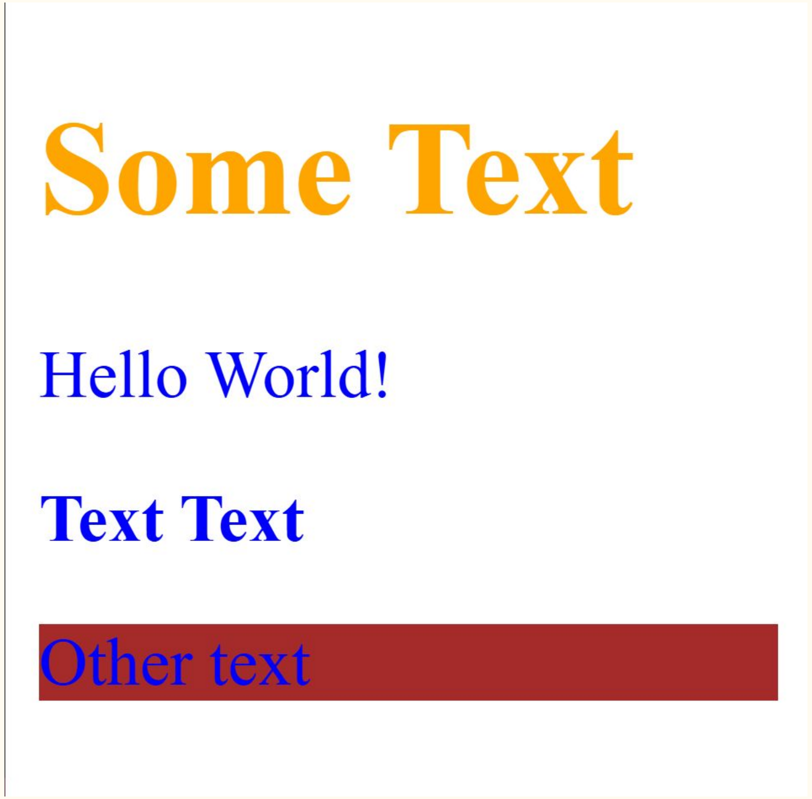
Overview - CSS

Source: https://developer.mozilla.org/en-US/docs/Web/CSS

HTML defines the structure of a web page but not how it is presented, how it’s **styled**.

Cascading Style Sheets (CSS) is a stylesheet language used to describe how a document written in HTML is presented. CSS describes how elements *should* be rendered on screen.

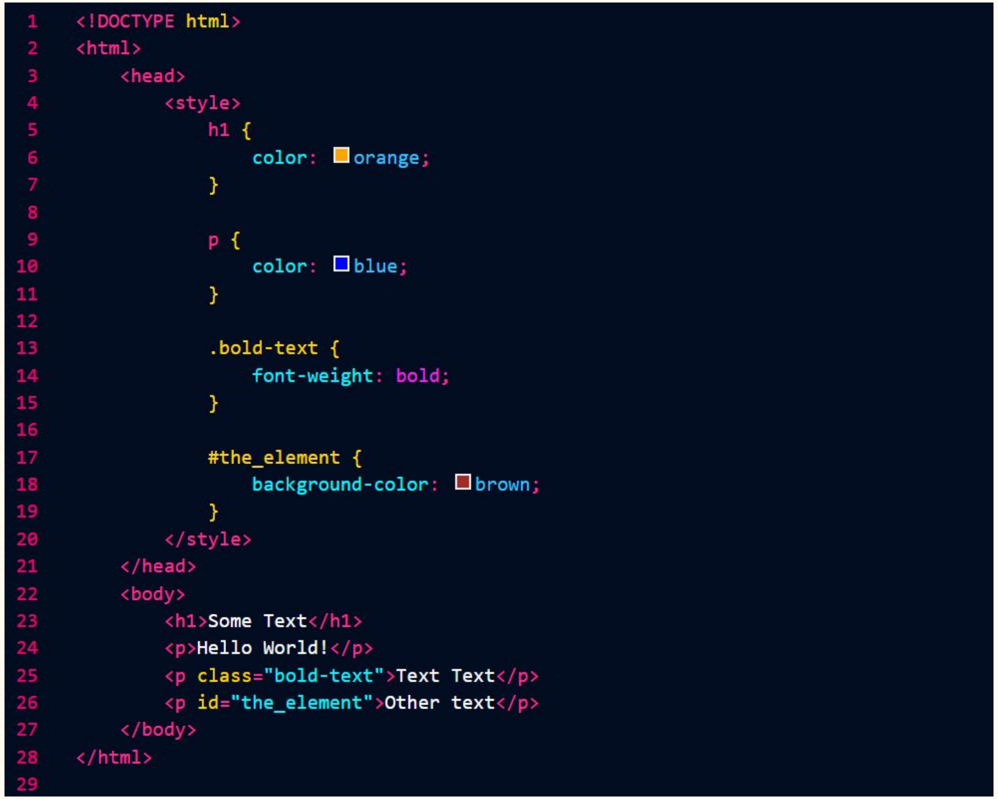
Overview - CSS


Some Text

Hello World!

Text Text

Other text

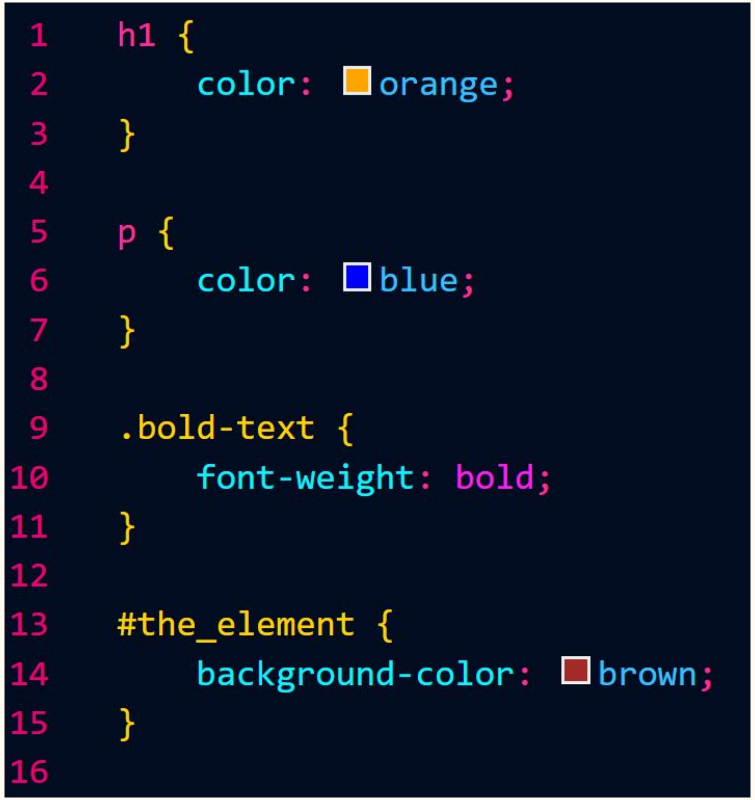

```html
<!DOCTYPE html>
<html>
    <head>
        <style>
            h1 {
                color: orange;
            }

            p {
                color: blue;
            }

            .bold-text {
                font-weight: bold;
            }

            #the_element {
                background-color: brown;
            }
        </style>
    </head>
    <body>
        <h1>Some Text</h1>
        <p>Hello World!</p>
        <p class="bold-text">Text Text</p>
        <p id="the_element">Other text</p>
    </body>
</html>
```

Overview - CSS



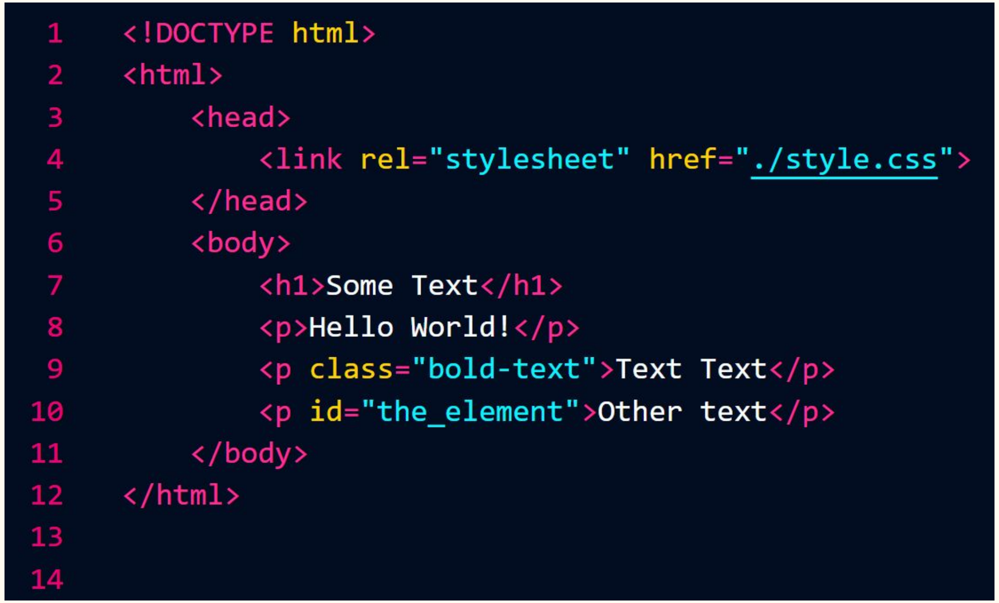
```css
h1 {
    color: 🟧orange;
}

p {
    color: 🟦blue;
}

.bold-text {
    font-weight: bold;
}

#the_element {
    background-color: 🟥brown;
}

```



```html
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="./style.css">
    </head>
    <body>
        <h1>Some Text</h1>
        <p>Hello World!</p>
        <p class="bold-text">Text Text</p>
        <p id="the_element">Other text</p>
    </body>
</html>


```

Overview - JavaScript

Source: https://developer.mozilla.org/en-US/docs/Web/JavaScript

JavaScript (JS) is an interpreted programming language initially created as a scripting language for Web pages.

A JavaScript (or ECMAScript) engine is the component that executes JavaScript code.

It utilize the **Document Object Model (DOM)** to have a representation of a web page in memory.

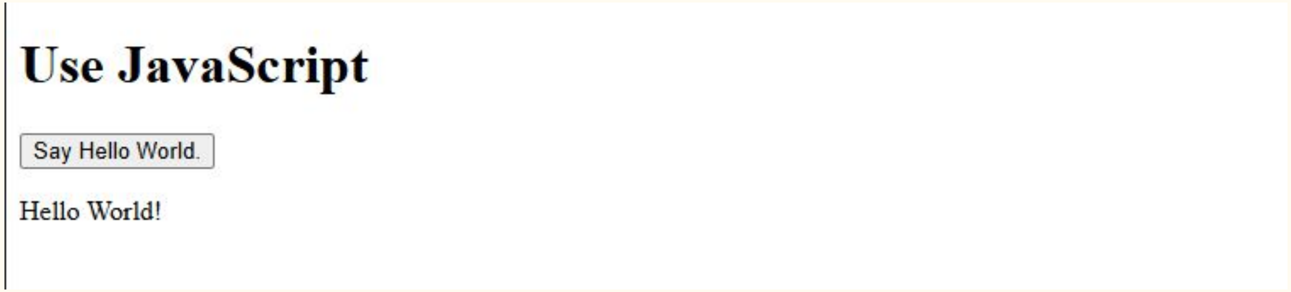
The DOM represents a document with a tree where each node contains objects. DOM methods allow JavaScript to access the tree and interact with the objects.

Overview - JavaScript

```html
<!DOCTYPE html>
<html>
    <body>
        <h1>Use JavaScript</h1>

        <button type="button" onclick="document.getElementById('target').innerHTML = 'Hello World!'">
            Say Hello World.
        </button>

        <p id="target"></p>
    </body>
</html>
```

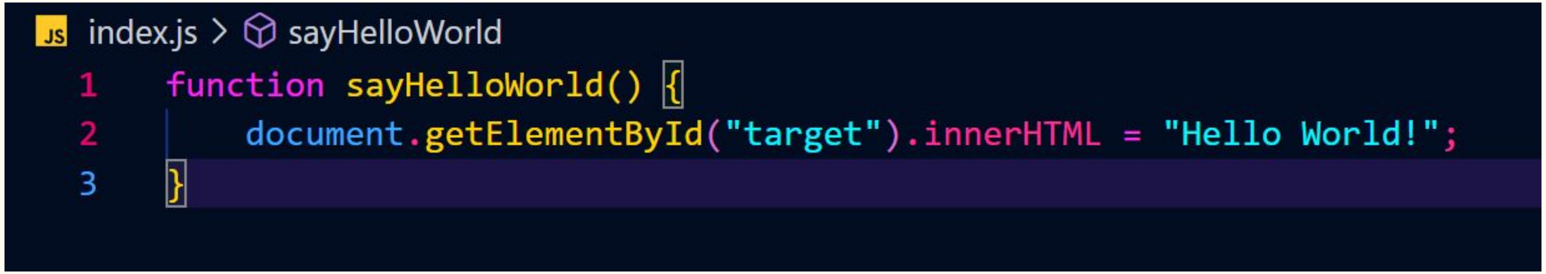


Use JavaScript

Say Hello World.

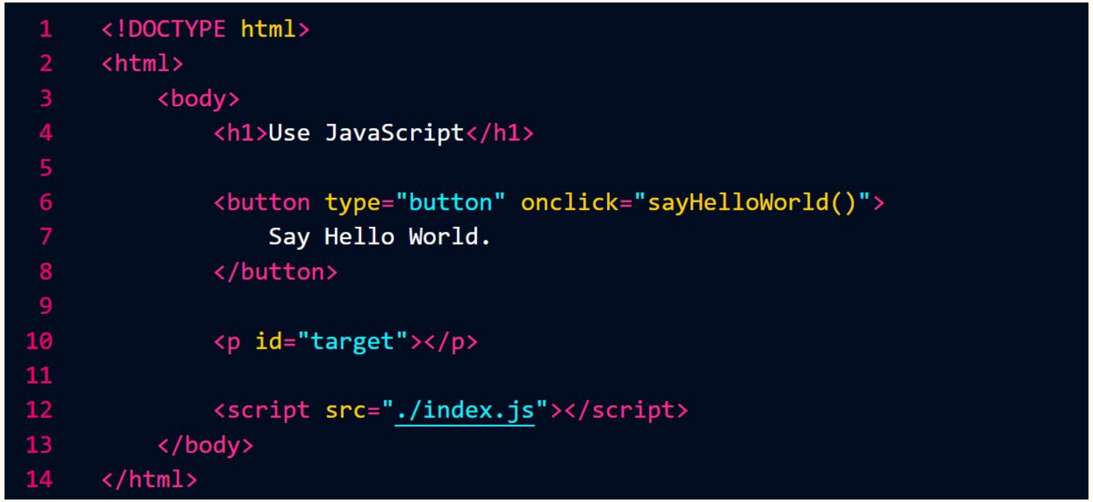
Hello World!

Overview - JavaScript



index.js > sayHelloWorld

```javascript
function sayHelloWorld() {
    document.getElementById("target").innerHTML = "Hello World!";
}
```



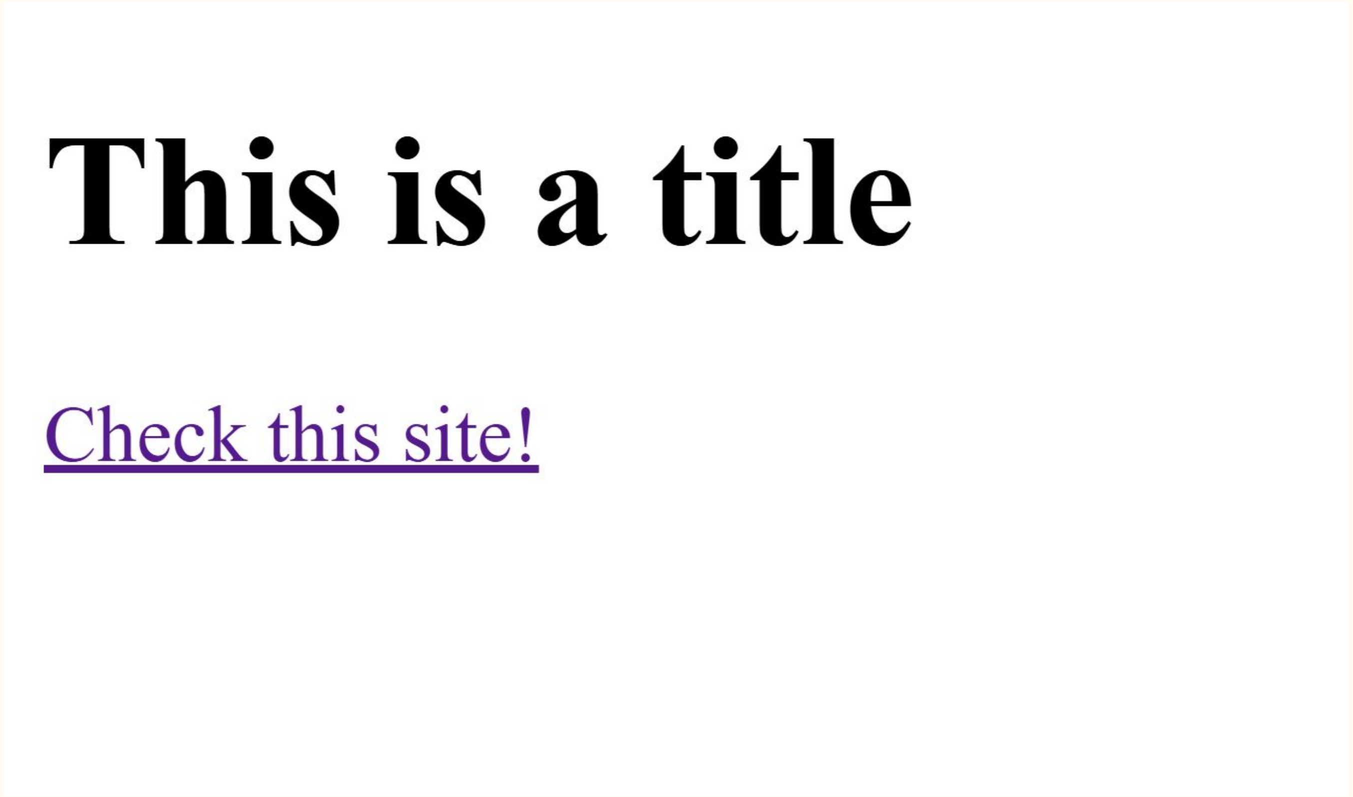
```html
<!DOCTYPE html>
<html>
    <body>
        <h1>Use JavaScript</h1>

        <button type="button" onclick="sayHelloWorld()">
            Say Hello World.
        </button>

        <p id="target"></p>

        <script src="./index.js"></script>
    </body>
</html>
```

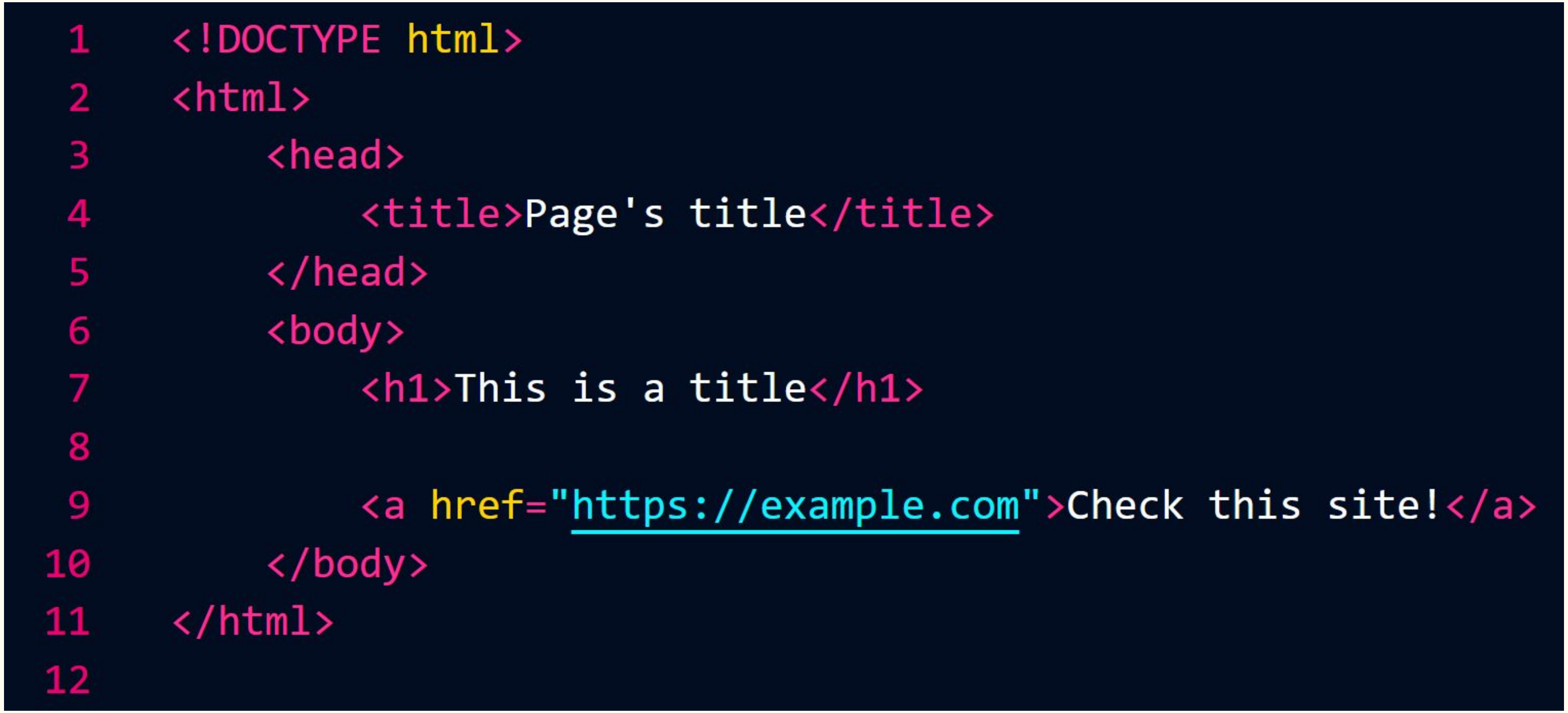
Overview - DOM



This is a title

Check this site!

Overview - DOM

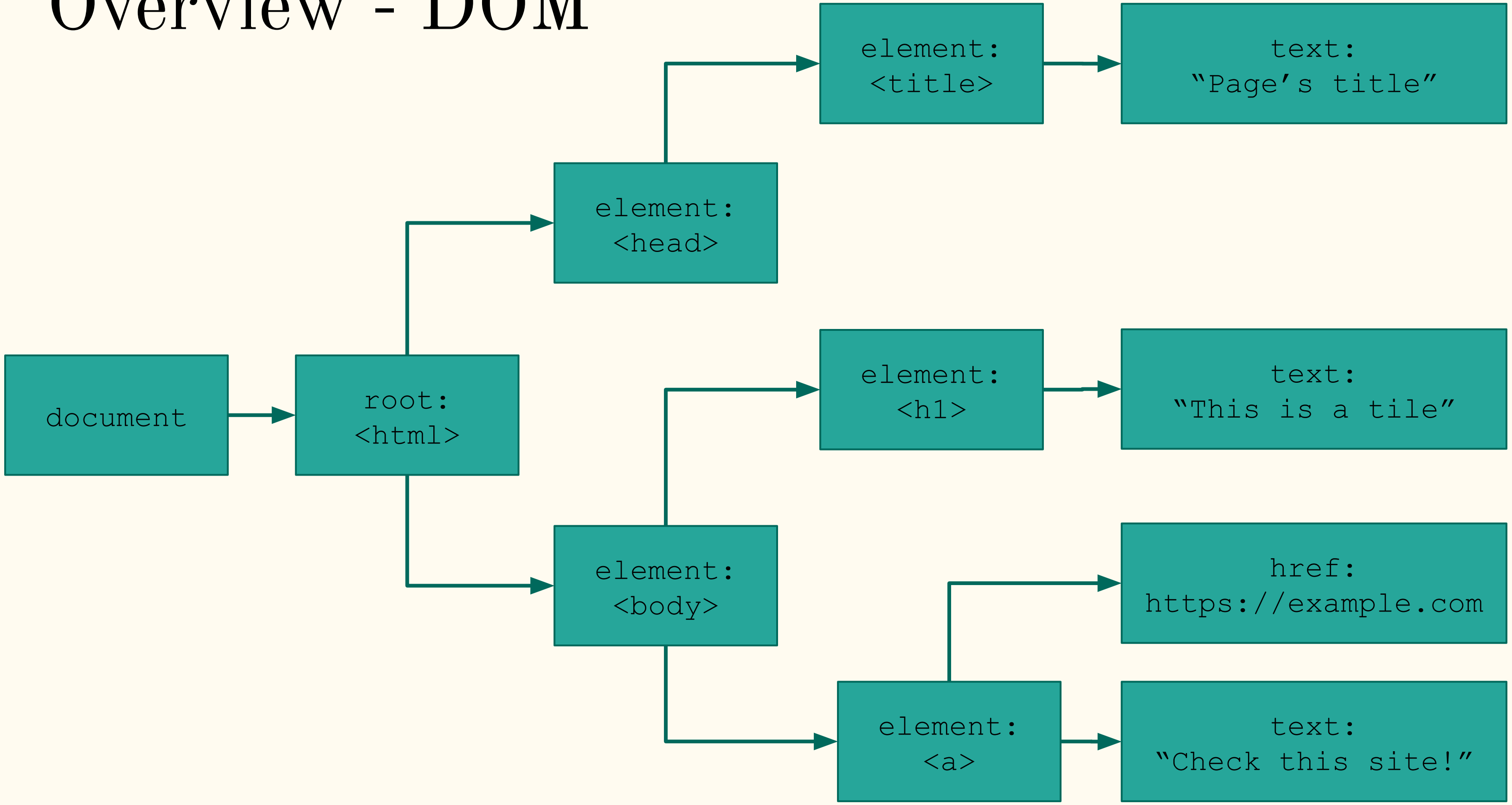


```html
<!DOCTYPE html>
<html>
    <head>
        <title>Page's title</title>
    </head>
    <body>
        <h1>This is a title</h1>

        <a href="https://example.com">Check this site!</a>
    </body>
</html>

```

Overview - DOM



XSS

XSS - Overview

Source: https://portswigger.net/web-security/cross-site-scripting

Cross-Site Scripting (XSS) is a type of code injection that specifically targets Web browsers by injecting JavaScript code inside Web pages.

By injecting malicious JavaScript code in a Web page an attacker can: steal session cookies, informations inside the page, execute requests on behalf of the victim (CSRF).

XSSs happen in three ways:

- By **reflection**
- By **stores**
- **DOM** based

XSS - Reflected

Reflected XSSs occur when unsafe user input is returned on the response page, without any type of sanitization.

The name *reflected* derive from how they take place, that is when the value of an HTTP variable is included (reflected) on the response page.

If the reflected value is not sanitized then is possible to inject JavaScript code in the page.

XSS - Reflected

Code:

```
<?php
   echo ‘hello ’ . $_GET[‘name’];
?>
```

URL:

```
http://example.com/page.php?name=Bob
```

Response:

```
<!DOCTYPE html>
<html>
<body>hello Bob</body>
</html>
```

XSS - Reflected

`<script>alert(1)</script>`

URL:

```
http://example.com/page.php?name=%3Cscript%3Ealert%281%29%3C%2Fscript%3E
```

Response:

```
<!DOCTYPE html>

<html>

<body>hello <script>alert(1)</script></body>

</html>
```

XSS - Stored

Stored XSSs are similar to reflected XSSs. In this case, the malicious code is injected without user interaction.

A stored XSS takes place when the injected code is stored by the site and than reflected without sanitization.

The attacker will find a way to make the website store the payload and later, make the victim visit the page with infected data.

XSS - Stored

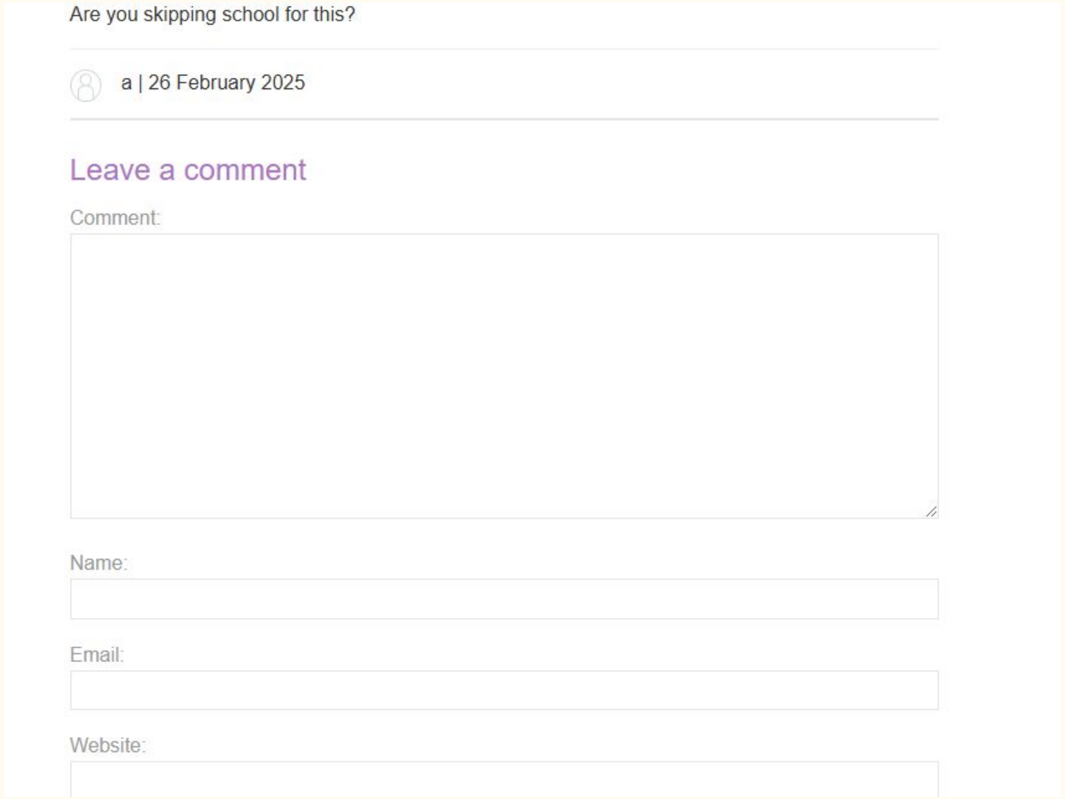
For example, in the case of a website that allows users to submit comments on blog posts, which are displayed to other users.

If a user leave a comment injected with malicious code, other users viewing that same comment will execute the injected code present inside the comment.

Try it:

- https://portswigger.net/web-security/cross-site-scripting/stored/lab-html-context-nothing-encoded

XSS - Stored



Are you skipping school for this?

a | 26 February 2025

Leave a comment

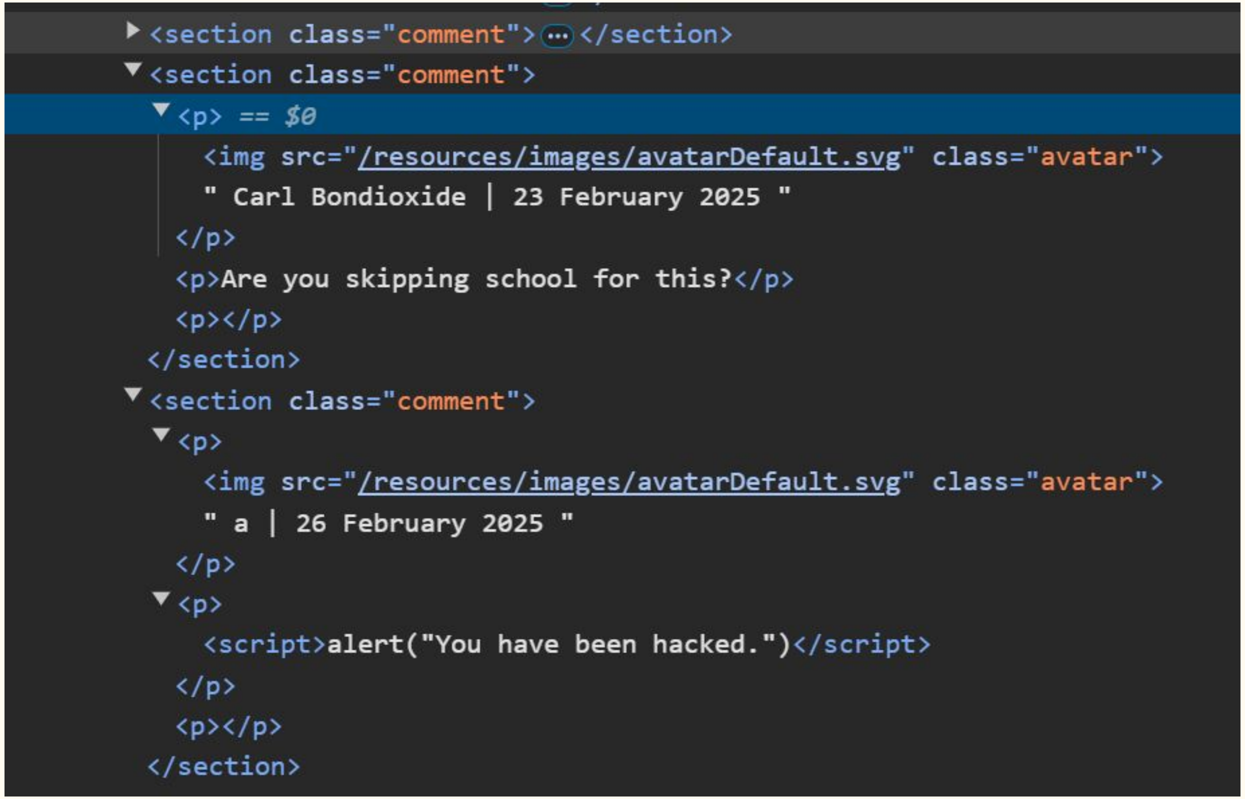
Comment:

Name:

Email:

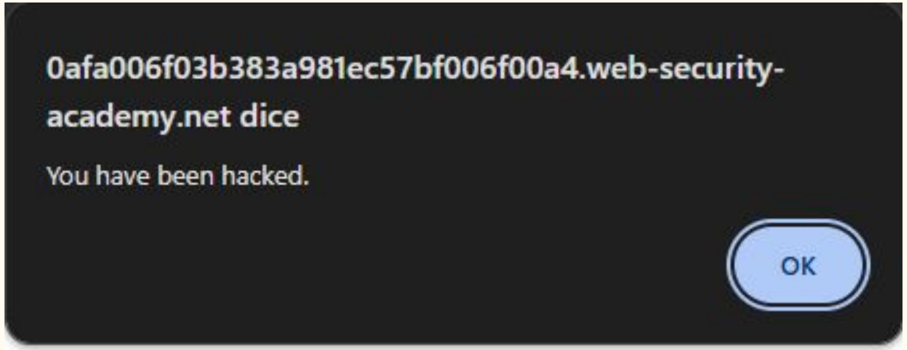
Website:

XSS - Stored



```
▶<section class="comment">⋯</section>
▼<section class="comment">
  ▼<p> == $0
      <img src="/resources/images/avatarDefault.svg" class="avatar">
      " Carl Bondioxide | 23 February 2025 "
    </p>
    <p>Are you skipping school for this?</p>
    <p></p>
  </section>
▼<section class="comment">
  ▼<p>
      <img src="/resources/images/avatarDefault.svg" class="avatar">
      " a | 26 February 2025 "
    </p>
  ▼<p>
      <script>alert("You have been hacked.")</script>
    </p>
    <p></p>
  </section>
```

XSS - Stored



0afa006f03b383a981ec57bf006f00a4.web-security-academy.net dice

You have been hacked.

XSS - Stored

Databases are not the only ones that store information sent by the client:

- https://portswigger.net/research/practical-web-cache-poisoning

XSS - Homeworks

- https://xss-game.appspot.com/
- https://zixem.altervista.org/XSS/
- https://xss.challenge.training.hacq.me/

CSRF

CSRF - Overview

Cross-Site Request Forgery (CSRF) is a type of attack that occurs when a victim’s Web browser perform an unwanted request on a trusted site for which the user has access.

CSRF are similar to SSRF but applied to client side technology.

XSSs can lead to CSRFs in some cases.

CSRF - Example

Imagine a bank that lets users send money to each other. Suppose that the request sent to the server looks like:

```
http://bank.com/transact?to=user2&money=1000
```

An attacker could replace *user2* with its account name and send the link to the victim.

If the victim clicks on the link while it’s logged in, then the transaction would start.

CSRF - Anti-CSRF token

Source: https://developer.mozilla.org/en-US/docs/Web/Security/Practical_implementation_guides/CSRF_prevention#anti-csrf_tokens

In the previous example, the server didn't know if the request was sent by the user performing the action on the bank website or by clicking on a link.

In order to prevent untrusted links from performing actions on a website an **anti-CSRF token** is required by the server.

This prevents CSRF attacks by requiring a secret, unique and random token on all the requests that cause destructive changes.

Anti-CSRF tokens are different from cookies, they are not stored and sent automatically during a request.

CSRF - Anti-CSRF token

```
<form action=”/transact” method=”POST”>
    <label></label>
    <input type=”text” name=”to” value=”Username” required>
    <input type=”number” name=”money” value=”10” required>
    <input type=”hidden” name=”csrf” value=”q398cNpryAqr29G38cA2a” required>

    <button class=”btn” type=”submit”>Send Money</button>
</form>
```

```
POST /transact HTTP/1.1
Host: bank.com
Content-Length: 70
Content-Type: application/x-www-from-urlencoded
Cookie: session=b9812aefa90cd390
X-CSRF-Token: q398cNpryAqr29G38cA2a

to=user2&money=1000&csrf=q398cNpryAqr29G38cA2a
```

JavaScript required

CSRF - Try it

- https://ctf.cyberchallenge.it/challenges#challenge-27
- https://ctf.cyberchallenge.it/challenges#challenge-129
- https://ctf.cyberchallenge.it/challenges#challenge-140

The End