Document Object Model (DOM)

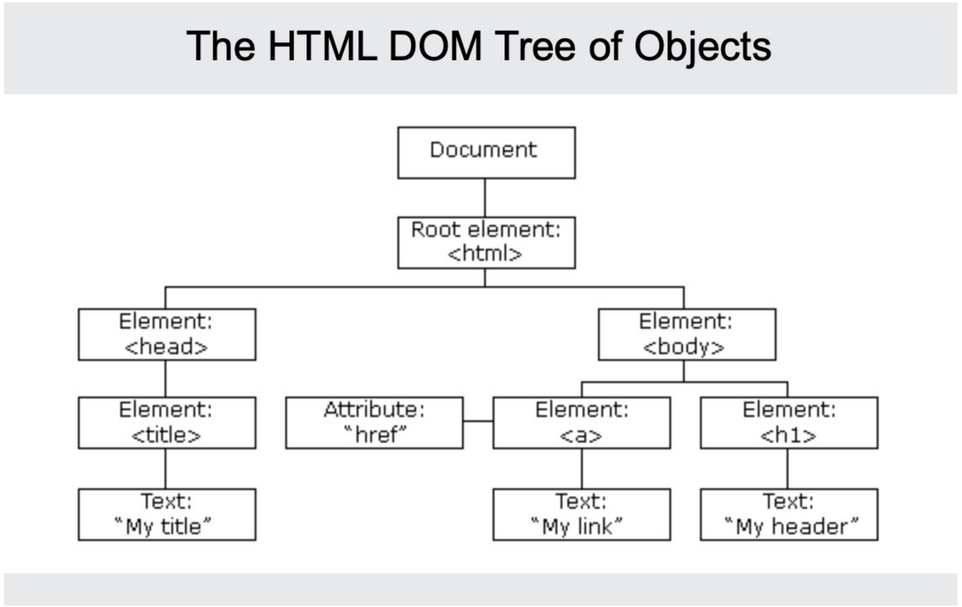
Selin

What is it and what does it do?

- A web API used to build websites
- W3C standard which provides a structured representation of HTML documents.
- Enables programs and scripts to “dynamically” access and update the content, structure, and style.
- 3 different parts:
    - Core DOM (for all document types)
    - XML DOM
    - HTML DOM

In short…

“The HTML DOM is a standard for how to get, change, add, or delete HTML elements.” - W3 Schools



- Each node of the tree corresponds to a part (attribute, text, element, etc.)

Why DOM?

1. Dynamic Web Development
2. Event Handling
    a. Interactive user interface
3. Accessibility and Search Engine Optimization?
    a. Indexing for SEO — Why is Google getting worse?
4. Single Page Applications
    a. No full page reload

“The DOM is not a programming language, but without it, the JavaScript language wouldn't have any model or notion of web pages, HTML documents, SVG documents, and their component parts.” - Mozilla Developer

DOM and Javascript

- DOM can be used with any programming language, but is most commonly used with JS.
- Fundamental data types:
    - Document: root object
    - Node: every object within the document is a node of some kind
    - Element: based on Node
    - NodeList: array of elements
    - Attr: also a type of node, has a special interface for attributes
    - NamedNodeMap

Dynamic WebDev by Event Handling

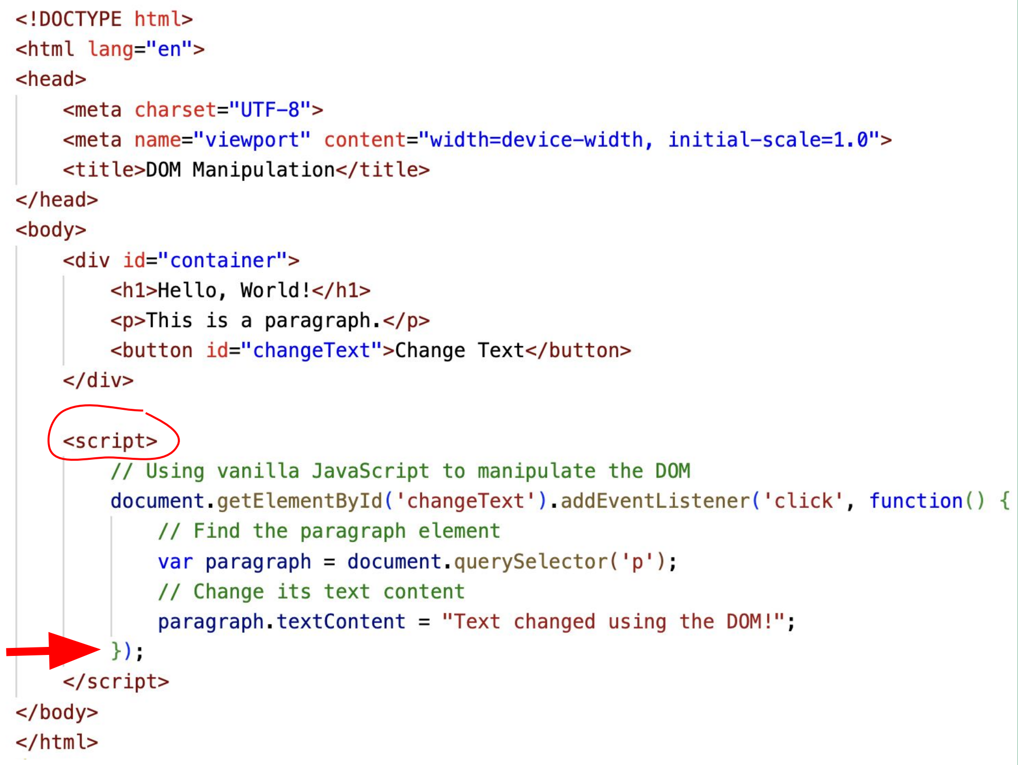
Event handlers are used to change the appearance of a website following user clicks or keystrokes, page reloads, browser resizing, form submission, error occurrences, video playing/pausing.

Common ones:

```
element.addEventListener('click', function(event)

element.addEventListener('mouseover', function(event)

element.addEventListener('mouseout', function(event)

element.addEventListener('keydown', function(event)

element.addEventListener('submit', function(event)

element.removeEventListener()

onClick()
```

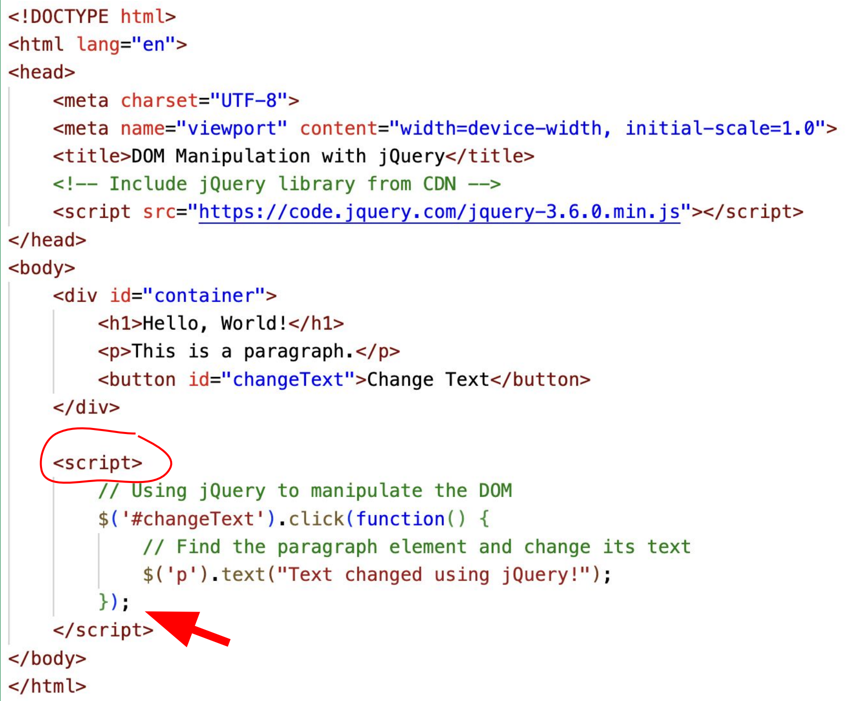
Code Example

Javascript



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DOM Manipulation</title>
</head>
<body>
    <div id="container">
        <h1>Hello, World!</h1>
        <p>This is a paragraph.</p>
        <button id="changeText">Change Text</button>
    </div>

    <script>
        // Using vanilla JavaScript to manipulate the DOM
        document.getElementById('changeText').addEventListener('click', function() {
            // Find the paragraph element
            var paragraph = document.querySelector('p');
            // Change its text content
            paragraph.textContent = "Text changed using the DOM!";
        });
    </script>
</body>
</html>
```

JQuery



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DOM Manipulation with jQuery</title>
    <!-- Include jQuery library from CDN -->
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
    <div id="container">
        <h1>Hello, World!</h1>
        <p>This is a paragraph.</p>
        <button id="changeText">Change Text</button>
    </div>

    <script>
        // Using jQuery to manipulate the DOM
        $('#changeText').click(function() {
            // Find the paragraph element and change its text
            $('p').text("Text changed using jQuery!");
        });
    </script>
</body>
</html>
```

Evils of the Web

- Cross Site Scripting
    - **DOM-based XSS (type-0 XSS):** Exists client-side. DOM is manipulated by attacker and malicious code is executed in user's browser.
    - **Stored (persistent XSS)**: Exists server-side. Malicious script is stored in a database and executed to all users when the website is visited.
    - **Reflected:** Exists client-side. Malicious script is injected through web URLs. Once the script is executed, it redirects or dynamically displays malicious code.

Blue-bus example

Uh-Oh! What to do to avoid?

- Input validation, sanitization
- Be careful what you’re publicly sharing
    - document.location(), location.search(), etc
- Be aware of vulnerable methods

- Don’t click on weird links…

The following are some of the main sinks that can lead to DOM-XSS vulnerabilities:

```
document.write()
document.writeln()
document.domain
element.innerHTML
element.outerHTML
element.insertAdjacentHTML
element.onevent
```

The following jQuery functions are also sinks that can lead to DOM-XSS vulnerabilities:

```
add()
after()
append()
animate()
insertAfter()
insertBefore()
before()
html()
prepend()
replaceAll()
replaceWith()
wrap()
wrapInner()
wrapAll()
has()
constructor()
init()
index()
jQuery.parseHTML()
$.parseHTML()
```

References

More detail on DOM & JS: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

https://chat.openai.com/

https://www.linkedin.com/pulse/importance-dom-web-development-aniruddh-diwan/

On XSS:

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

https://owasp.org/www-community/attacks/DOM_Based_XSS

https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html