
# DOM Manipulation

# Goals

1. Access and target webpage elements using JavaScript.
2. Dynamically change content and style of elements
3. Respond to clicks, typing, and other user interactions.

# What Is DOM?

- DOM stands for Document Object Model
- It’s a programming interface that represents an HTML document as a structured hierarchy of objects
  - Each HTML element becomes an object (node) in a tree structure

# Why the DOM Exists

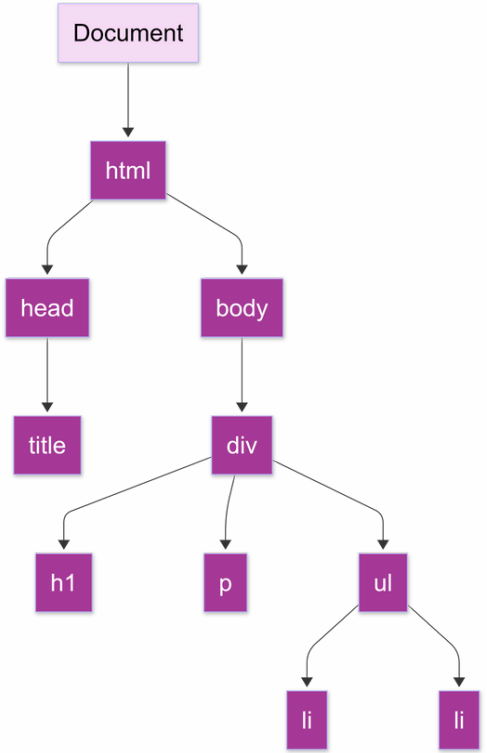
- HTML is static and cannot respond to user actions on its own
- The DOM enables JavaScript-driven interactivity after the page loads
- Allows dynamic updates such as:
  Changing text and content
  Updating styles and layout
  Adding or removing elements
- Makes it possible to respond to user events (clicks, typing, key presses)

# DOM Tree Structure

- The DOM represents a web page as a tree of nodes
- The document object sits at the top and represents the entire page
- Below it are main elements such as html, head, and body
- Elements are connected through parent, child, and sibling relationships
- This structure allows JavaScript to navigate and manipulate specific elements

# DOM Tree Structure

```
<html>
  <head>
    <title>Webpage Title</title>
  </head>
  <body>
    <div>
      <h1>This is Heading Tag</h1>
      <p>Some Text Content</p>
      <ul>
        <li>List Item</li>
      </ul>
    </div>
  </body>
</html>
```



# Nodes in the DOM

- Everything in the DOM is represented as a node
- Element nodes represent HTML tags (e.g., div, p)
- Text nodes contain the text inside elements
- Attribute nodes store element attributes (e.g., id, class)
- JavaScript treats nodes as objects with properties and methods that can be accessed or modified

# DOM vs HTML

| HTML | DOM |
| --- | --- |
| Static code | Dynamic representation of the page |
| Cannot be changed by JavaScript | Can be changed dynamically with JavaScript (text, styles, elements) |
| Static; page won’t respond to user actions | Interactive; responds to clicks, typing, and other events |

# Accessing the DOM with JavaScript

- The document object is the main entry point to the DOM
- It represents the entire web page in the browser
- All DOM manipulation begins by accessing document
- JavaScript uses document to locate, create, and modify elements

```
console.log(document);
```

# Selecting Elements

- DOM manipulation starts by selecting elements
- JavaScript must first find an element before changing it
- Elements can be selected using:
    - ID
    - Class
    - Tag name
    - CSS selectors

# Selecting by ID

- An ID uniquely identifies a single element
- getElementById returns the element object
- Allows direct access to content, styles, and events

```
const title = document.getElementById("title")
title.textContent = "New Heading"
```

# Selecting with querySelector

- Uses familiar CSS selector syntax
- Can select by tag, class, or ID
- Always returns the first matching element

```
document.querySelector("p")
document.querySelector(".box")
document.querySelector("#title")
```

# Selecting Multiple Elements

- querySelectorAll selects all matching elements
- Returns a NodeList

```
const items = document.querySelectorAll(".item");

items.forEach(item => {
  console.log(item.textContent);
});
```

# Modifying Text Content

- JavaScript can change page text dynamically

```
const para = document.querySelector("p")
para.textContent = "This text was changed using JavaScript"
```

# Modifying HTML Content

## innerHTML allows inserting HTML elements

```
element.innerHTML = "<strong>Hello World</strong>";
```

# Modifying Attributes

- Attributes control element behavior and information
- JavaScript can add, read, or remove attributes dynamically

```
element.setAttribute("class", "active")
element.getAttribute("href")
element.removeAttribute("disabled")
```

# Modifying Styles with JavaScript

THRIVE

**Styles can be updated using the style property**

```
box.style.backgroundColor = "lightblue";
box.style.padding = "10px";
```

# Creating Elements

JavaScript can create elements dynamically

```
const div = document.createElement("div");
div.textContent = "New Element";
```

# Adding Elements to the DOM

To make a created element visible, it must be appended to the DOM.

```
document.body.appendChild(div);
```

# Removing Elements

Elements can be removed directly or through their parent.

```html
<ul id="list">
  <li id="item1">Apple</li>
  <li id="item2">Oranage</li>
</ul>
```

```javascript
const item = document.getElementById("item2");
item.remove();
```

# Removing Elements

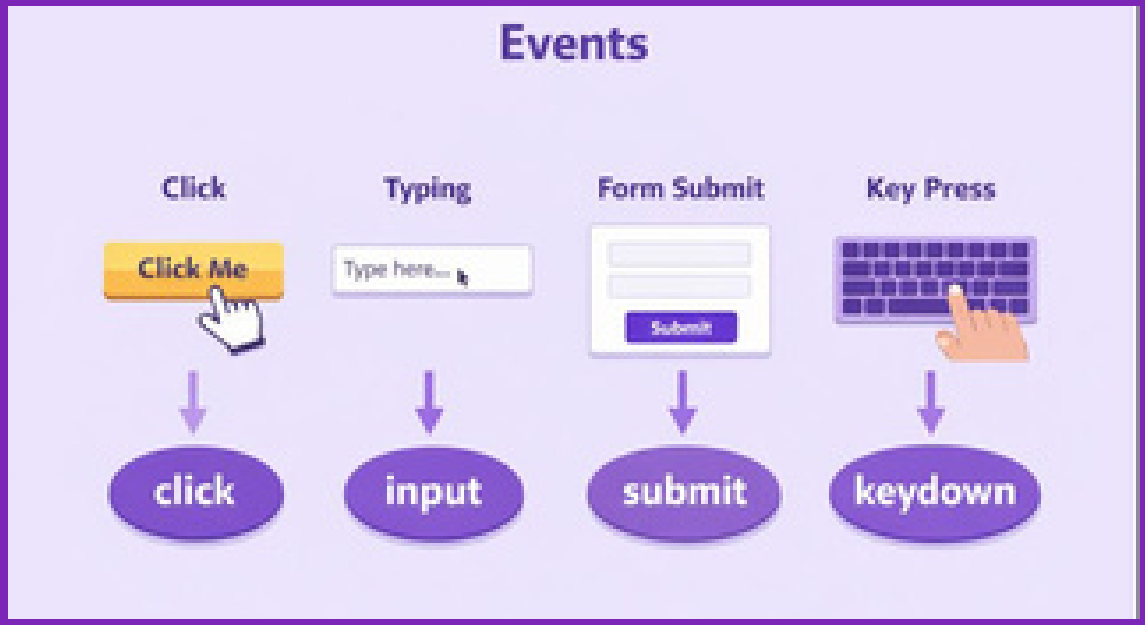
## Parent Removing a Child

```
const parent = document.getElementById("list");
const child = document.getElementById("item2");

parent.removeChild(child);
```

# Introduction to Events

Events represent actions such as clicks, typing, or form submissions.



# Event Listeners

Event listeners allow JavaScript to respond to user actions.

- A callback function runs whenever the specified event occurs.

```
<button id="btn">Click Me</button>
```

```
document.getElementById("btn")
  .addEventListener("click", () => {
    alert("Button clicked!");
  });
```

# Basic Syntax of Event Listeners

- element → the HTML element
- event → action to listen for (e.g. "click")
- callback → function that runs when the event happens

```
element.addEventListener("event", callback);
```

# Event Object

Each event provides an event object containing details about what happened.

```
btn.addEventListener("click", (event) => {
  console.log(event.target);
});
```

# Input Events

Input events allow JavaScript to react as users type.

```
input.addEventListener("input", () => {
  output.textContent = input.value;
});
```

# Exercises

1. Select all <li> elements from a page and log them to the console.
2. Change the text of a heading and update the background color of a div using JavaScript, applying both text and style manipulation concepts.
3. Dynamically create a button and add it to the page using JS.
4. Create a button interaction where clicking the button changes paragraph text.

# Summary & Wrap-Up

- DOM manipulation begins with document
- Elements are selected, modified, created, or removed
- Events allow pages to react to user actions