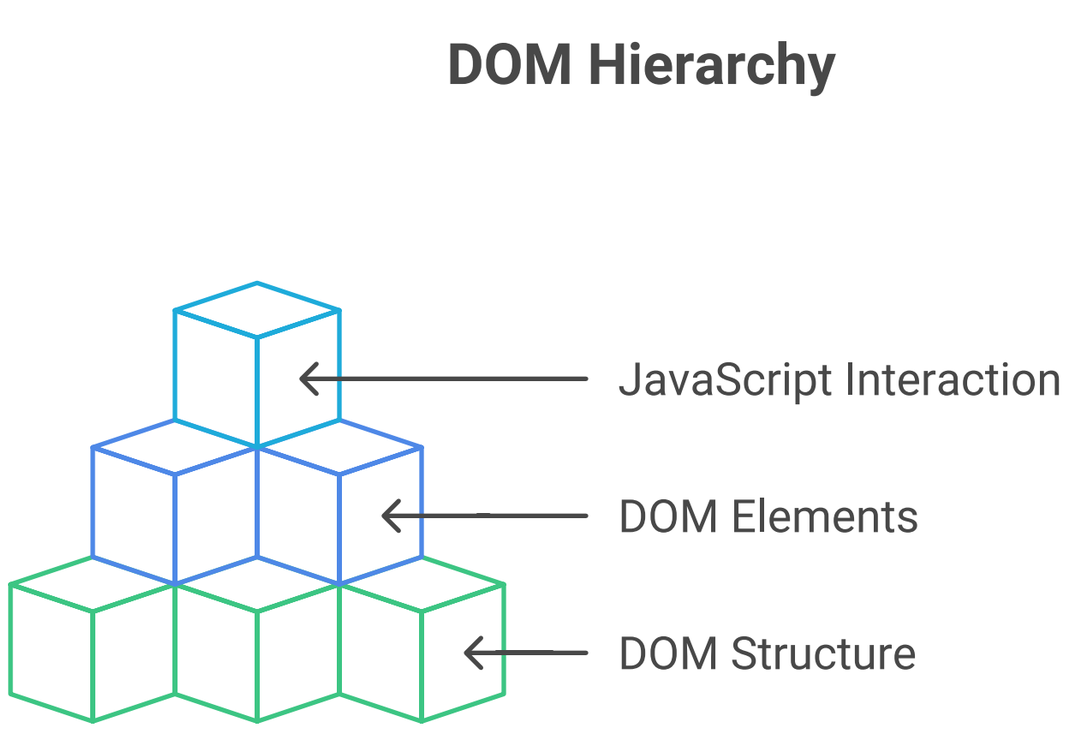
DOM Manipulation: An IT Industry Manual with Real-Time Scenarios

This document serves as a practical guide to DOM (Document Object Model) manipulation using HTML, CSS, and JavaScript, tailored for IT professionals. It provides a foundational understanding of the DOM, explores common manipulation techniques, and illustrates their application through real-time, basic scenarios encountered in web development. This manual aims to equip developers with the essential skills to dynamically modify web page content, structure, and styling, enhancing user experience and interactivity.

Understanding the DOM

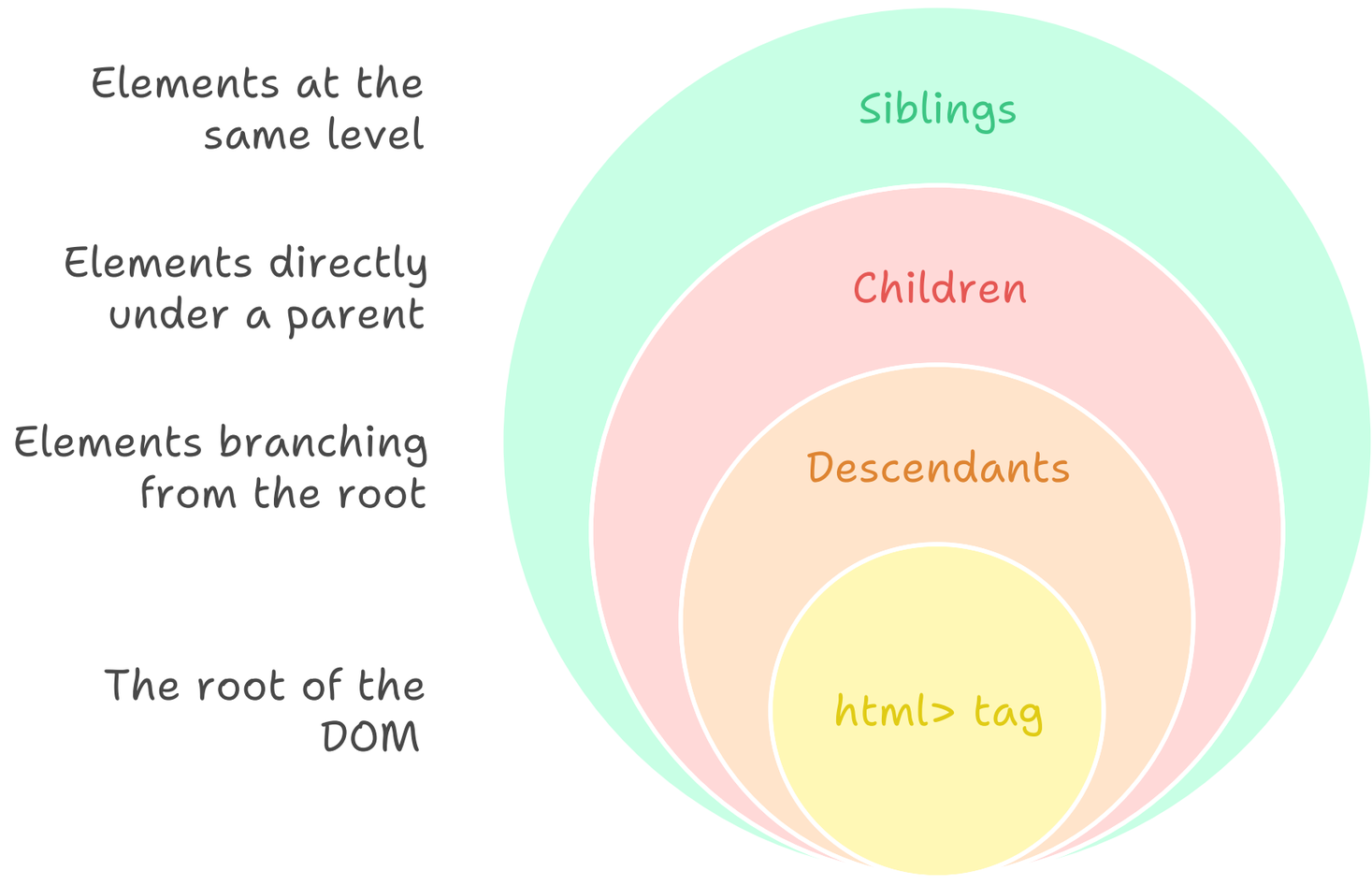
The Document Object Model (DOM) is a programming interface for HTML and XML documents. It represents the page as a tree-like structure, where each element, attribute, and text node is an object. JavaScript uses the DOM to access and manipulate the content, structure, and style of a web page.



The DOM Tree

Imagine an HTML document as a family tree. The <html> tag is the root, and all other elements are its descendants. Each element can have children (other elements or text) and siblings (elements at the same level).

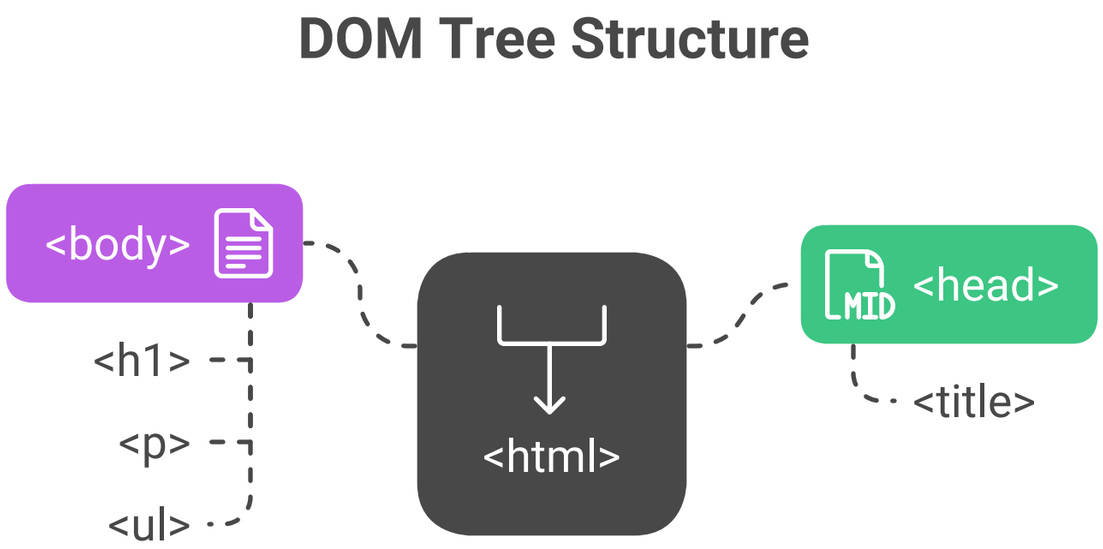
DOM Structure

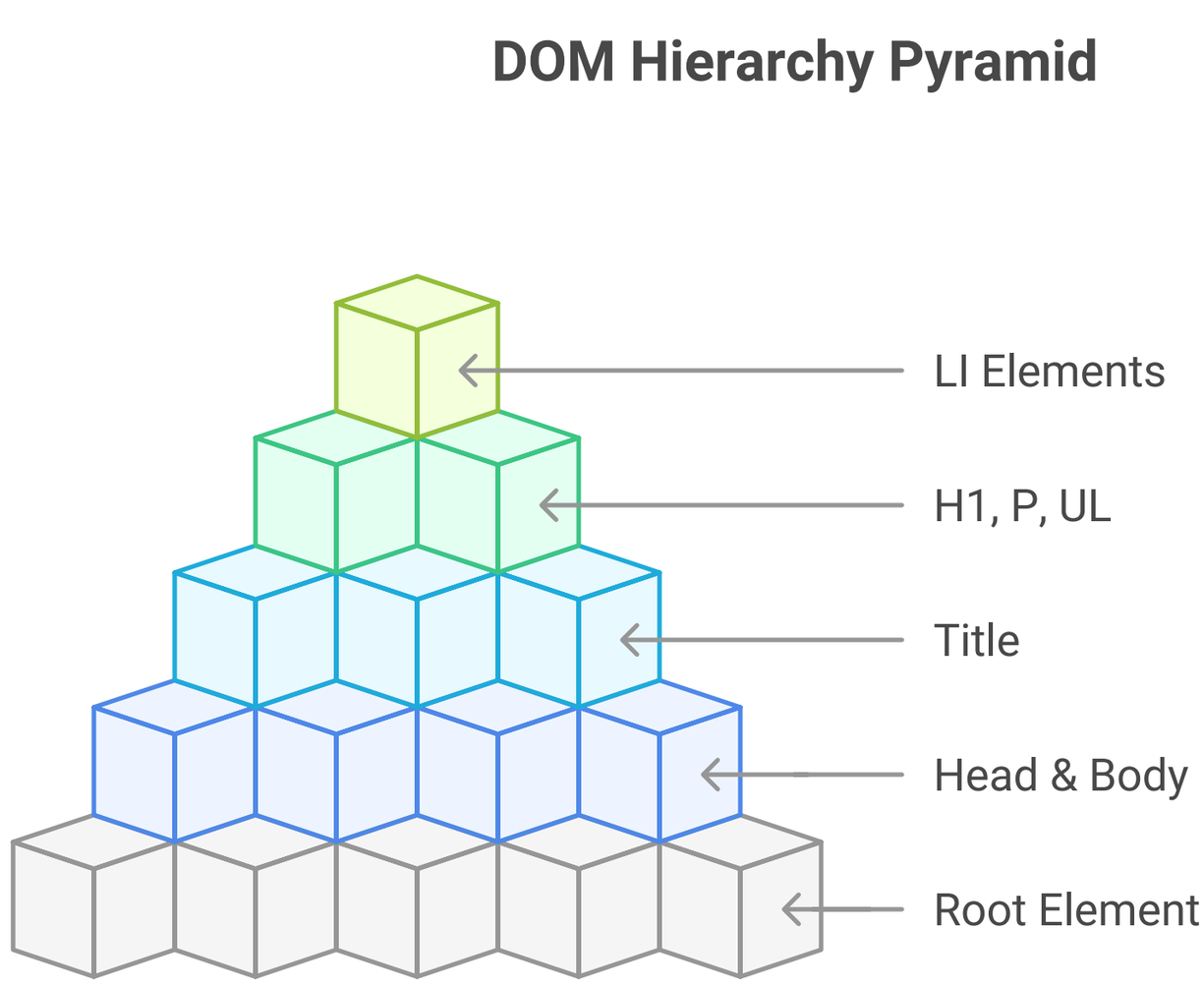



```
<!DOCTYPE html>
<html>
<head>
  <title>My Web Page</title>
</head>
<body>
  <h1>Welcome!</h1>
  <p>This is a paragraph.</p>
  <ul id="myList">
    <li>Item 1</li>
    <li>Item 2</li>
  </ul>
</body>
</html>
```

In this example:

- <html> is the root element.
- <head> and <body> are children of <html>.
- <title> is a child of <head>.
- <h1>, <p>, and <ul> are children of <body>.
- elements are children of <ul>.





Accessing DOM Elements

JavaScript provides several methods to access elements within the DOM:

- document.getElementById(id): Retrieves an element by its unique id attribute. This is the fastest and most efficient method when you know the ID.

> ```javascript

```
const myList = document.getElementById("myList");
```

```
console.log(myList); // Output: <ul id="myList">...</ul>
```

```

- document.getElementsByClassName(className)**:** Retrieves all elements with a specific class name.  Returns an HTMLCollection (an array-like object).

```html
<p class="highlight">This is highlighted.</p>
```

```
<p class="highlight">This is also highlighted.</p>
```

```
<script>
```

```
const highlighted = document.getElementsByClassName("highlight");
```

```
console.log(highlighted); // Output: HTMLCollection(2) [p.highlight,
p.highlight]
```

```
</script>
```

```

- document.getElementsByTagName(tagName)**:** Retrieves all elements with a specific tag name. Returns an HTMLCollection.

```javascript
const paragraphs = document.getElementsByTagName("p");
```

```
console.log(paragraphs); // Output: HTMLCollection(2) [p, p.highlight]
```

```

- document.querySelector(selector)**:** Retrieves the *first* element that matches a specified CSS selector.  This is more versatile than getElementById but can be slower if used repeatedly.
```

```javascript

```

```
const firstHighlight = document.querySelector(".highlight");
```

```
console.log(firstHighlight); // Output: <p class="highlight">This is
highlighted.</p>
```

```

```

- document.querySelectorAll(selector)**:** Retrieves *all* elements that match a specified CSS selector. Returns a NodeList (similar to an HTMLCollection).

```javascript

```

```
const allHighlights = document.querySelectorAll(".highlight");
```

```
console.log(allHighlights); // Output: NodeList(2) [p.highlight, p.highlight]
```

```

```

Manipulating DOM Elements

Once you have access to an element, you can manipulate its properties, content, and style.

Modifying Content

- element.innerHTML**:** Gets or sets the HTML content of an element. Use with caution as it can be a security risk if you're injecting user-provided content.

```javascript

```

```
const heading = document.querySelector("h1");
```

```
heading.innerHTML = "Updated Welcome!";
```

```

```

- element.textContent**:** Gets or sets the text content of an element. Safer than innerHTML for injecting user-provided content.

```javascript
```

```javascript
const paragraph = document.querySelector("p");
```

```javascript
paragraph.textContent = "This paragraph has been updated.";
```

```
```

- element.setAttribute(attributeName, attributeValue)**:** Sets the value of an attribute on an element.

```javascript
```

```javascript
const link = document.createElement('a');
```

```javascript
link.setAttribute('href', 'https://www.example.com');
```

```javascript
link.textContent = 'Visit Example';
```

```javascript
document.body.appendChild(link);
```

```
```

- element.getAttribute(attributeName)**:** Gets the value of an attribute on an element.

```javascript
```

```javascript
const link = document.querySelector('a');
```

```javascript
const href = link.getAttribute('href');
```

```javascript
console.log(href); // Output: https://www.example.com
```

```
```

Modifying Styles

- element.style.propertyName**:** Sets the inline style of an element.

```javascript
const heading = document.querySelector("h1");

heading.style.color = "blue";

heading.style.fontSize = "2em";
```

- element.classList.add(className): Adds a class to an element.

```javascript
const paragraph = document.querySelector("p");

paragraph.classList.add("highlight");
```

- element.classList.remove(className): Removes a class from an element.

```javascript
paragraph.classList.remove("highlight");
```

- element.classList.toggle(className): Toggles a class on an element (adds if it's not there, removes if it is).

```javascript
paragraph.classList.toggle("highlight");
```

Creating and Appending Elements

- document.createElement(tagName): Creates a new HTML element.

```javascript
```

```
const newParagraph = document.createElement("p");
```

```
newParagraph.textContent = "This is a new paragraph.";
```

```
```

- element.appendChild(newElement)**:** Appends a new element as a child of an existing element.

```javascript
```

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

```
```

- element.removeChild(childElement)**:** Removes a child element from an element.

```javascript
```

```
document.body.removeChild(newParagraph);
```

```
```

- element.parentNode**:** Gets the parent node of an element.

```javascript
```

```
const listItem = document.querySelector('li');
```

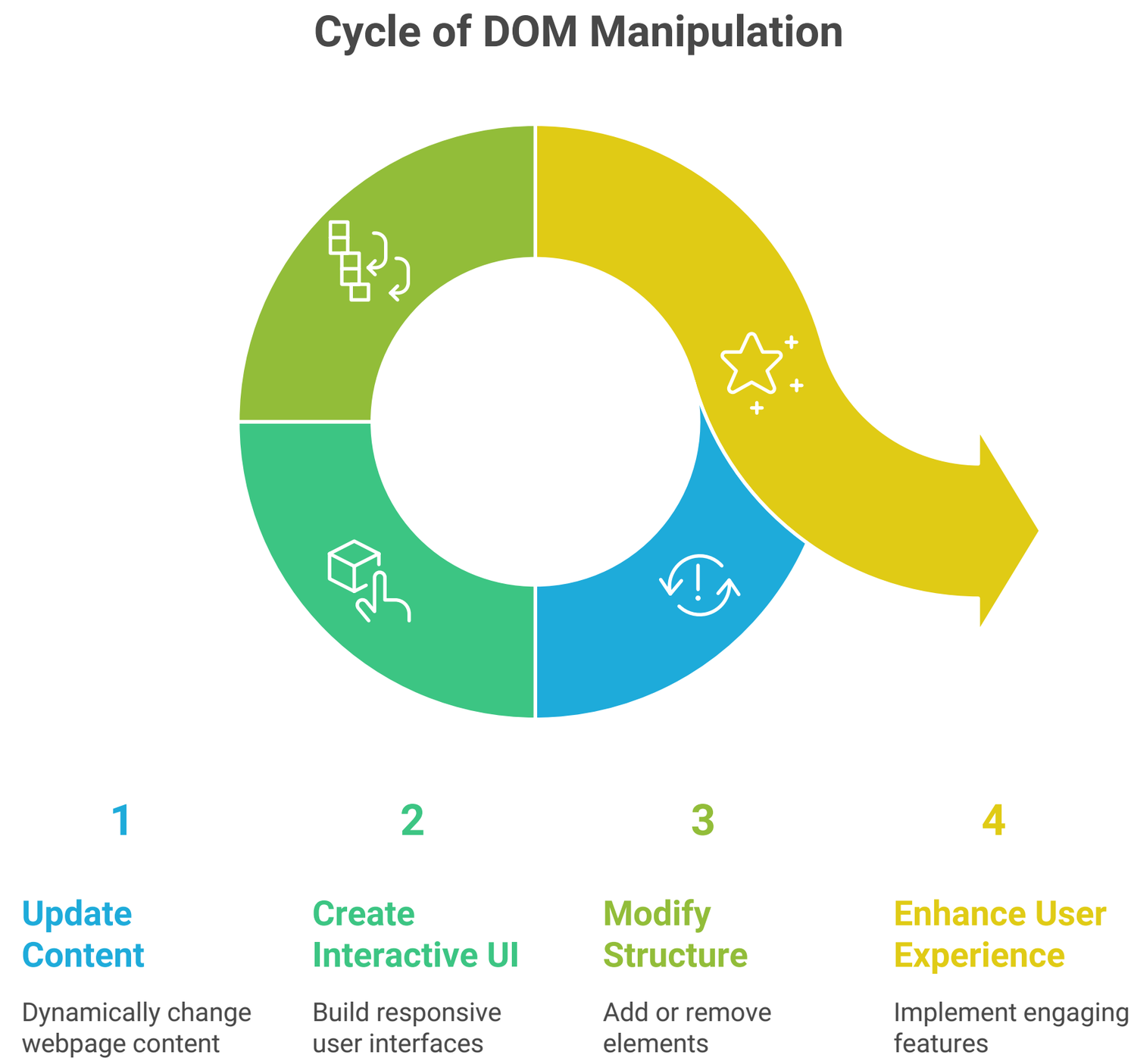
```
const parentList = listItem.parentNode;
```

```
console.log(parentList); // Output: <ul id="myList">...</ul>
```

```
```

Real-Time Basic Scenarios

Here are some common scenarios where DOM manipulation is used:



1. **Displaying a welcome message based on the time of day:**

```javascript
```

```
const greetingElement = document.getElementById("greeting");
```

```
const now = new Date();
```

```
const hour = now.getHours();
```

```
let greeting;
```

```
if (hour < 12) {
```

```
  greeting = "Good morning!";
```

```
} else if (hour < 18) {
```

```
  greeting = "Good afternoon!";
```

```
} else {
```

```
  greeting = "Good evening!";
```

```
}
```

```
greetingElement.textContent = greeting;
```

```
```

```
```html
```

```
<div id="greeting"></div>
```

```
```

2. **Adding a new item to a list:**

```
```javascript
```

```
const list = document.getElementById("myList");
```

```
const newItemText = "Item 3";
```

```
const newItem = document.createElement("li");
```

```
newItem.textContent = newItemText;
```

```
list.appendChild(newItem);
```

```

3. **Hiding/Showing an element based on a button click:**

```javascript

```
const toggleButton = document.getElementById("toggleButton");

```
const hiddenElement = document.getElementById("hiddenElement");

```
toggleButton.addEventListener("click", function() {

```
  if (hiddenElement.style.display === "none") {

```
    hiddenElement.style.display = "block";

```
  } else {

```
    hiddenElement.style.display = "none";

```
  }

```
});

```

```

```html

```
<button id="toggleButton">Toggle Element</button>

```
<div id="hiddenElement" style="display: none;">This element is hidden.</div>

```

4. **Dynamically updating content from an API:** (Simplified example)

```javascript
```
```

```
// Assume you have data from an API call:
```

```
const apiData = { title: "New Title", description: "
```