**JAVA SCRIPT**

**Javascript Fundamentals**

Javascript is a versatile programming language that is commonly used for web development. It allows developers to create dynamic and interactive content on websites.

**1. Variables and Data Types**

**Variables:** Used to store and manipulate data.

var x = 10;

let y = "Hello";

const PI = 3.14;

**Data Types:** JavaScript has several data types

1. **Primitive types:** number, string, boolean, null, undefined.
2. **Complex types:** object, array, function.

**2. Operators:**

**Arithmetic Operators:** +, -, \*, /, %.

**Comparison Operators:** ==, ===, !=, !==, >, <, >=, <=.

**Logical Operators:** && (and), || (or), ! (not).

**3. Control Flow:**

**Conditionals:** if, else if, else.

if (condition) {

  // code to be executed if the condition is true

} else {

  // code to be executed if the condition is false

}

**Loops:** for, while, do-while.

for (let i = 0; i < 5; i++) {

  // code to be repeated

}

while (condition) {

  // code to be repeated while the condition is true

}

**4. Functions:**

**Declaration:**

function add(a, b) {

  return a + b;

}

**Anonymous Functions (Function Expressions):**

const multiply = function(x, y) {

  return x \* y;

};

**Arrow Functions (ES6):**

const square = (num) => num \* num;

**5. Arrays:** Ordered collections of values.

const fruits = ['apple', 'orange', 'banana'];

**6. Objects:** Unordered collections of key-value pairs.

const person = {

  name: 'John',

  age: 25,

  isStudent: false

};

**7. DOM Manipulation:**

JavaScript can be used to interact with the Document Object Model (DOM) to update and modify HTML elements on a web page.

document.getElementById('myElement').innerHTML = 'New Content';

**8. Events:**

JavaScript can respond to user actions, such as clicks or keypresses.

button.addEventListener('click', function() {

  // code to be executed when the button is clicked

});

**9. Asynchronous JavaScript:**

Handling asynchronous operations using callbacks, promises, and async/await.

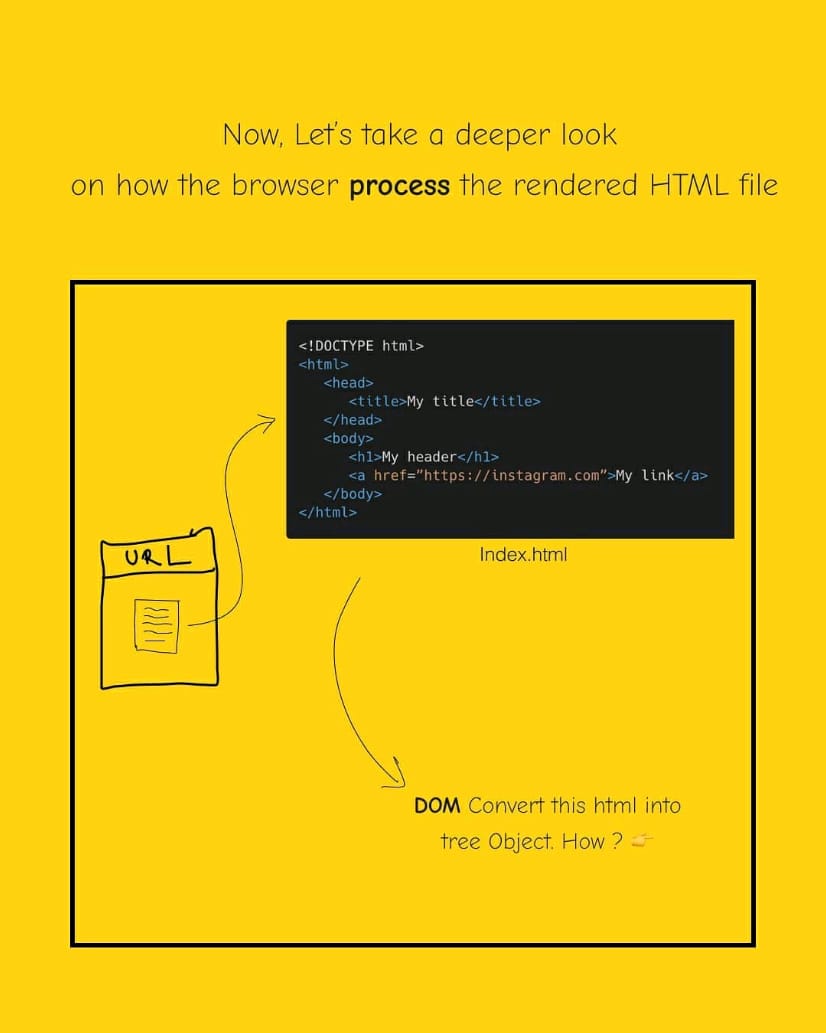
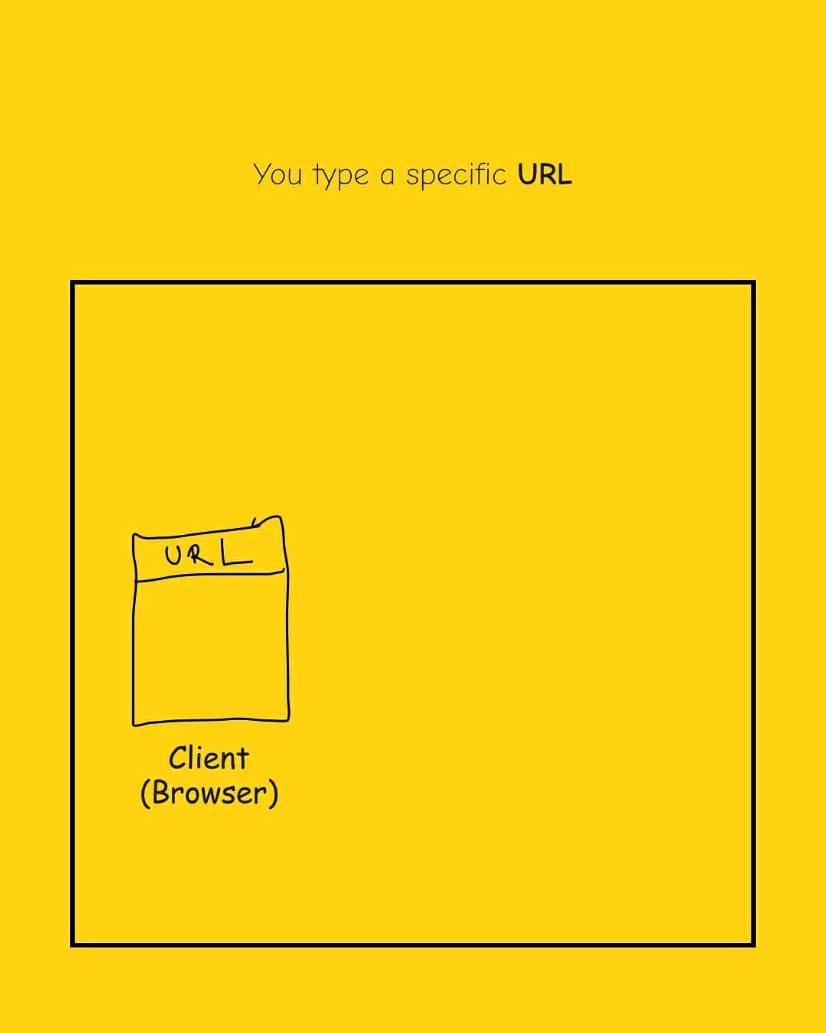
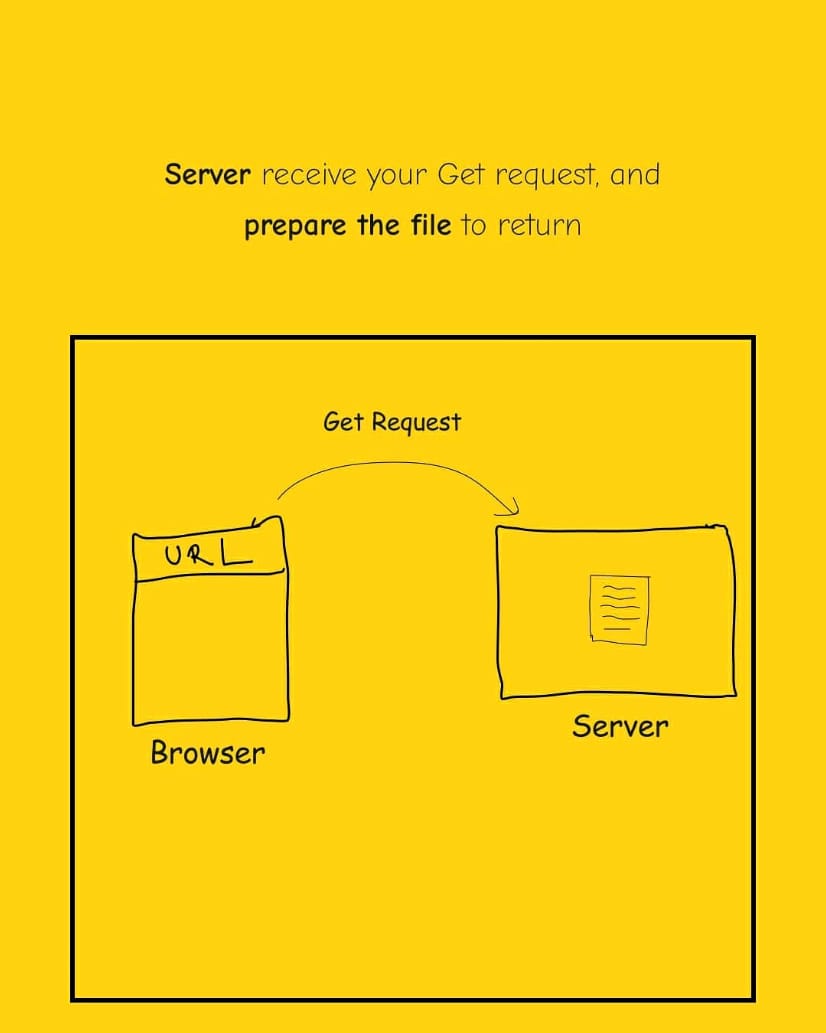
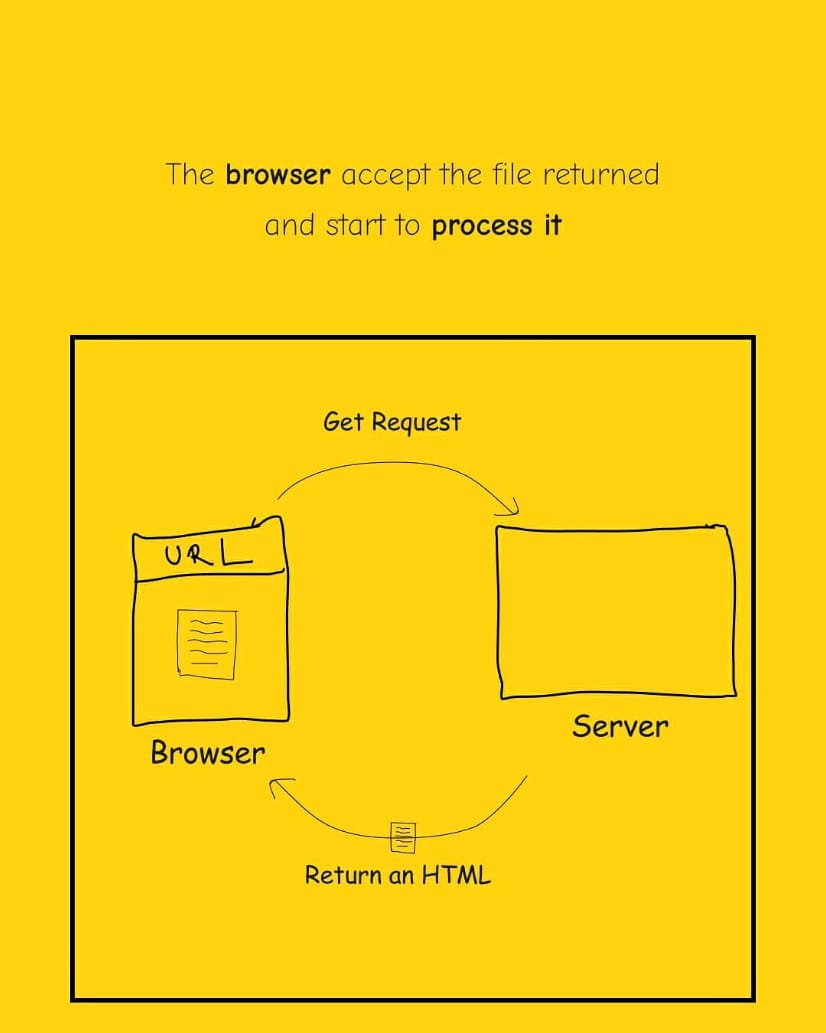
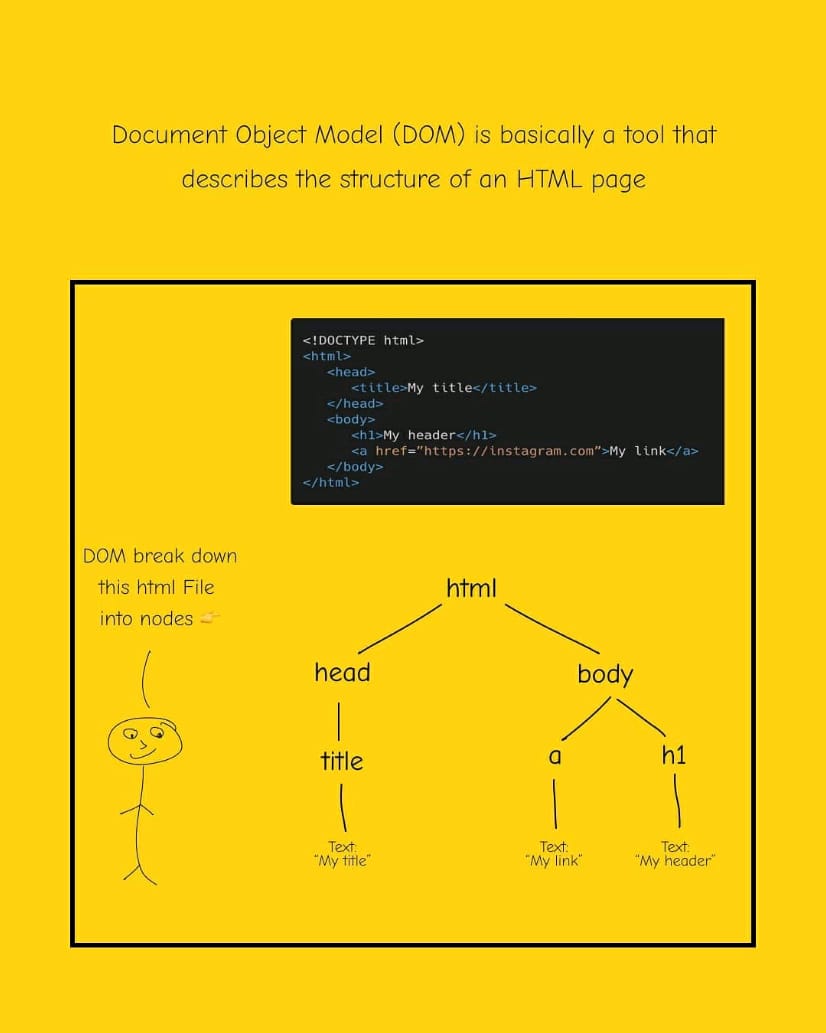
fetch('https://api.example.com/data')

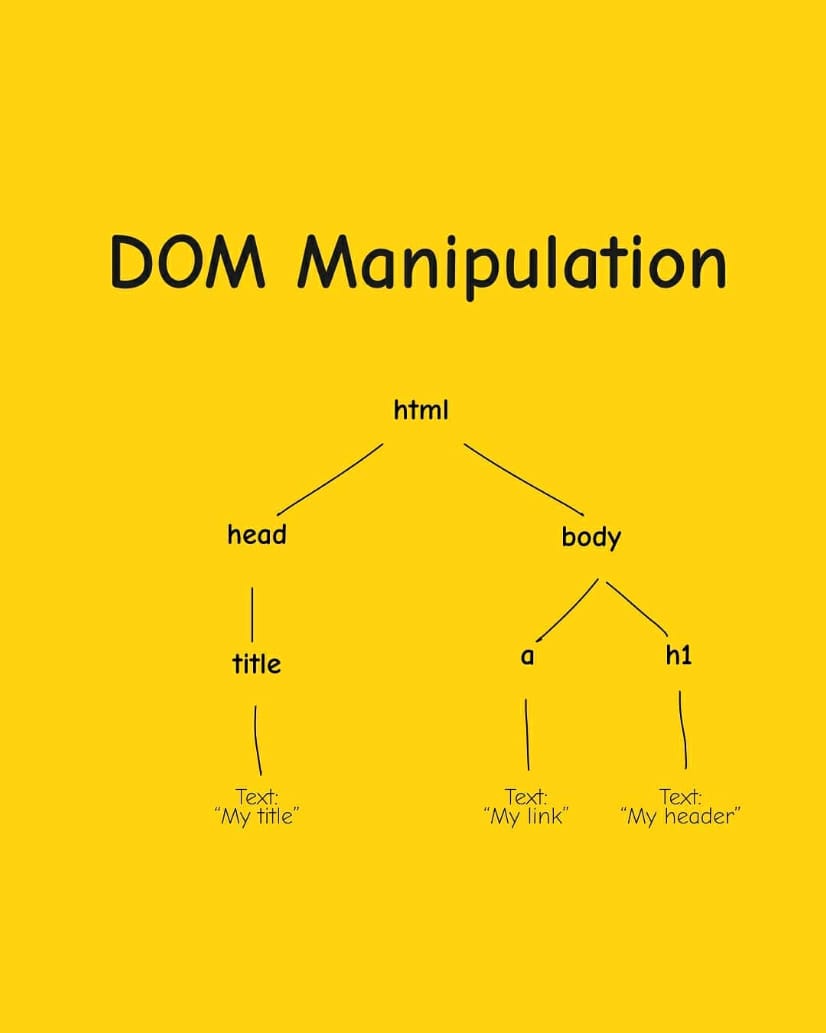
  .then(response => response.json())

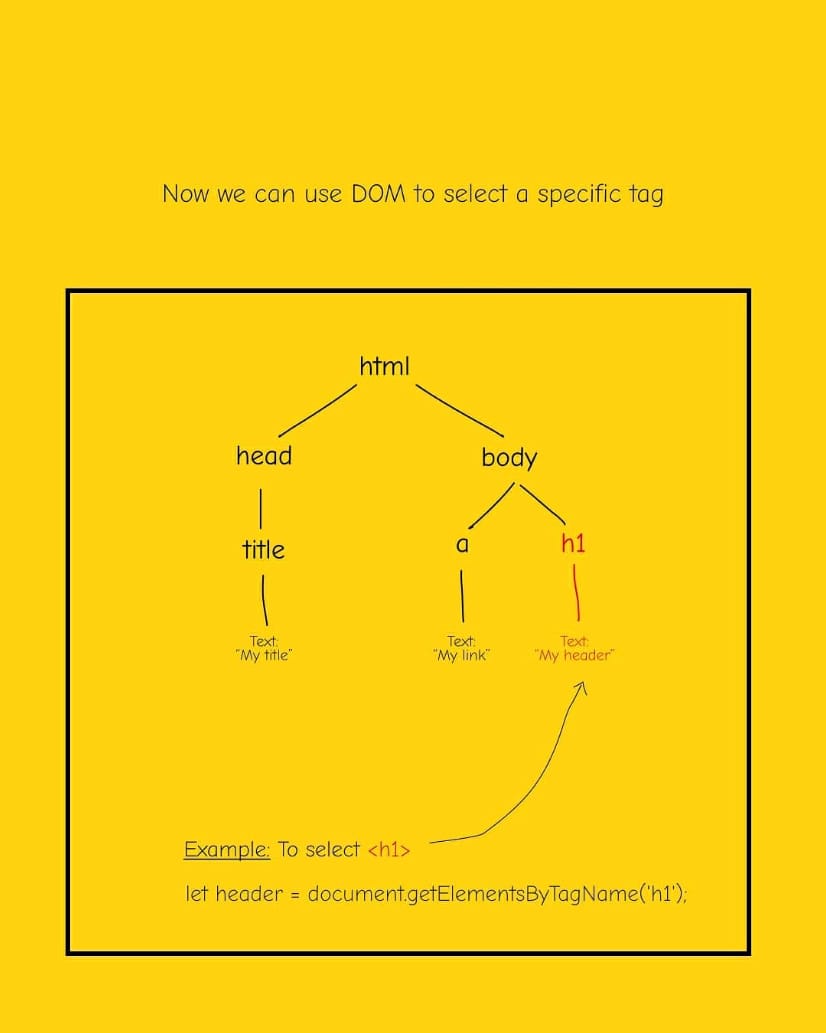
  .then(data => console.log(data))

  .catch(error => console.error(error));

**Document Object Model(DOM)**

 In JavaScript, documents typically refer to the Document Object Model (DOM). The DOM is a programming interface for web documents, and it represents the structure of a document as a tree of objects. This tree structure allows JavaScript to interact with and manipulate the content of a web page.





**Here are some commonly used DOM methods to select HTML Elements:**

**1.getElementById(id):**

Retrieves an HTML element with the specified ID.

var element = document.getElementById('myId');

**2.getElementsByClassName(className):**

Retrieves a collection of HTML elements with the specified class name.

var elements = document.getElementsByClassName('myClass');

**3.getElementsByTagName(tagName):**

Retrieves a collection of HTML elements with the specified tag name.

var paragraphs = document.getElementsByTagName('p');

**4.querySelector(selector):**

Retrieves the first HTML element that matches the specified CSS selector.

var element = document.querySelector('.myClass');

**5.querySelectorAll(selector):**

Retrieves a NodeList containing all HTML elements that match the specified CSS selector.

var elements = document.querySelectorAll('p.myClass');

**DOM MANIPULATION**

**1.createElement(tagName):**

Creates a new HTML element with the specified tag name.

var newElement = document.createElement('div');

**1.appendChild(node):**

Appends a child node to the end of the list of children of a specified parent node.

parentElement.appendChild(newElement);

**2.removeChild(node):**

Removes a child node from the DOM.

parentElement.removeChild(childElement);

**3.setAttribute(name, value):**

Sets the value of an attribute on the specified element.

element.setAttribute('src', 'image.jpg');

**4.addEventListener(event, callback):**

Attaches an event listener to the specified element, calling the provided callback function when the event occurs.

element.addEventListener('click', function() {

// code to execute when the element is clicked

});

**5.innerHTML:**

Gets or sets the HTML content within an element.

element.innerHTML = 'New content';

**6.textContent:**

Gets or sets the text content within an element.

element.textContent = 'Text content';

These methods provide the foundation for manipulating the structure and content of HTML documents using JavaScript. Keep in mind that manipulating the DOM directly can sometimes be inefficient, especially for complex applications, so modern frameworks often abstract away some of this complexity.

**WEB ELEMENTS**

These methods and properties are frequently used in web development for dynamically manipulating the attributes and classes of HTML elements using JavaScript.

**1.getAttribute:**

getAttribute is a method that retrieves the value of a specified attribute on the element. If the attribute does not exist, it returns null.

**Example:**

var value = element.getAttribute('attributeName');

**2.setAttribute:**

setAttribute is a method that sets the value of a specified attribute on the element. If the attribute does not exist, it will be created.

**Example:**

element.setAttribute('attributeName', 'attributeValue');

**3.removeAttribute:**

removeAttribute is a method that removes a specified attribute from the element.

**Example:**

element.removeAttribute('attributeName');

**4.classList:**

Definition: classList is a property that returns a live DOMTokenList collection of the class attributes of the element.

**Example:**

var classes = element.classList;

**5.classList.add:**

Definition: classList.add is a method that adds one or more class names to the element. If the specified class already exists, it will be ignored.

**Example:**

element.classList.add('className');

**6.classList.remove:**

classList.remove is a method that removes one or more class names from the element.

**Example:**

element.classList.remove('className');

**7.classList.toggle:**

classList.toggle is a method that toggles the presence of a class on the element. If the class exists, it is removed; if it does not exist, it is added.

**Example:**

element.classList.toggle('className');

**DOM ELEMENT PROPERTIES**

These properties and methods are commonly used in DOM manipulation to navigate and interact with the structure of HTML documents using JavaScript.

**1.parentElement:**

parentElement is a property that returns the parent element node of the specified element.

**Example:**

var parent = element.parentElement;

**2.nextElementSibling:**

nextElementSibling is a property that returns the next sibling element node of the specified element.

**Example:**

var nextSibling = element.nextElementSibling;

**3.nextSibling:**

nextSibling is a property that returns the next sibling node (including text nodes and comment nodes) of the specified element.

**Example:**

var nextSibling = element.nextSibling;

**4.previousElementSibling:**

previousElementSibling is a property that returns the previous sibling element node of the specified element.

**Example:**

var prevSibling = element.previousElementSibling;

**5.previousSibling:**

previousSibling is a property that returns the previous sibling node (including text nodes and comment nodes) of the specified element.

**Example:**

var prevSibling = element.previousSibling;

**6.childNode:**

childNode is a property that returns the first child node of the specified element. It can include elements, text nodes, and comment nodes.

**Example:**

var firstChild = element.childNodes[0];

**7.children:**

children is a property that returns a live HTMLCollection of child elements of the specified element.

**Example:**

var childElements = element.children;