JavaScript

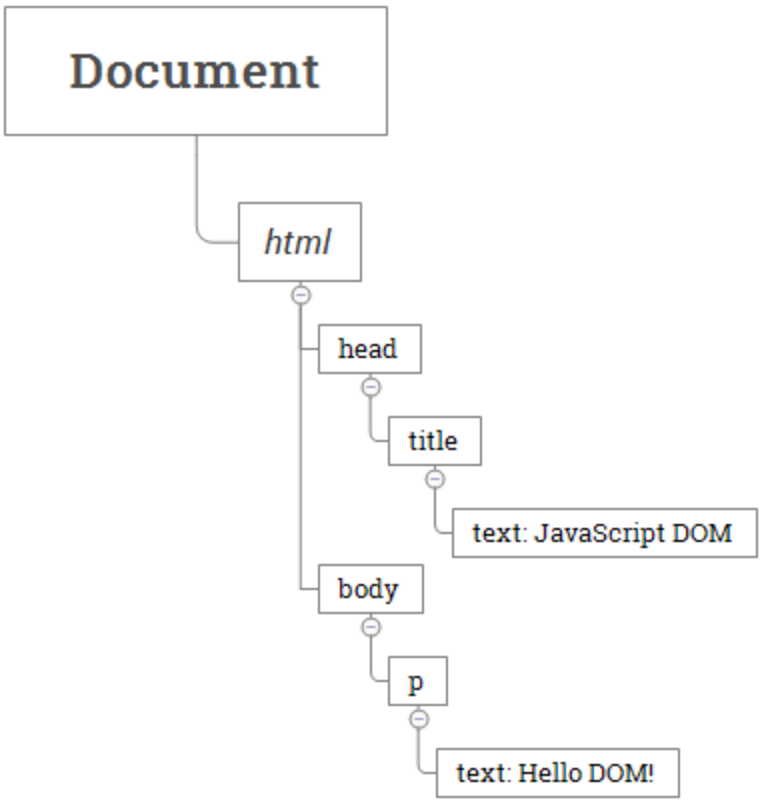
DOM Manipulation

The Document Object Model (DOM) is an application programming interface (API) for manipulating HTML documents.

The DOM represents an HTML document as a tree of nodes. The DOM provides functions that allow you to add, remove, and modify parts of the document effectively.

Document as a hierarchy of nodes

The DOM represents an HTML document as a hierarchy of nodes.



In this DOM tree, the document is the root node. The root node has one child node which is the <html> element. The <html> element is called the *document element*.

Each document can have only one document element. In an HTML document, the document element is the <html> element. Each markup can be represented by a node in the tree.

Node Types

Each node in the DOM tree is identified by a node type. JavaScript uses integer numbers to determine the node types. The following table illustrates the node type constants:

Constant	Value	Description
Node.ELEMENT_NODE	1	An Element node like <p> or .
Node.TEXT_NODE	3	The actual Text inside an Element or Attr.
Node.COMMENT_NODE	8	A Comment node, such as <!-- … -->.

To get the type of node, you use the nodeType property:

node.nodeType

Introduction to JavaScript getElementById() method

The getElementById() is a method of the document object that returns an Element object representing an HTML element with an id matching a specified string.

Here's the syntax of the getElementById() method:

const element = document.getElementById(id);

If the document has no element with the specified id, the getElementById() method returns null.

JavaScript getElementById() method example

Suppose you have a document with two p elements:

** **Hi, There!** **

** **JavaScript is fun.** **Code language: HTML, XML (xml)

The following code shows how to get the element with the id first:

const elem = document.getElementById("first");

JavaScript getElementsByName

Every element on an HTML document may have a name attribute:

<input type="radio" name="language" value="JavaScript">Code language: HTML, XML (xml)

Unlike the id attribute, multiple HTML elements can share the same value of the name attribute like this:

<input type="radio" name="language" value="JavaScript">

<input type="radio" name="language" value="TypeScript">

To get all elements with a specified name, you use the getElementsByName() method of the document object:

let elements = document.getElementsByName(name);Code language: JavaScript (javascript)

The getElementsByName() accepts a name which is the value of the name attribute of elements and returns a live NodeList of elements.

```
let rates = document.getElementsByName('rate');
        rates.forEach((rate) => {
            if (rate.checked) {
                output.innerText = `You selected: ${rate.value}`;
            }
        });
```

JavaScript getElementsByTagName

The getElementsByTagName() is a method of the document object or a specific DOM element.

The getElementsByTagName() method accepts a tag name and returns a live HTMLCollection of elements with the matching tag name in the order in which they appear in the document.

The following illustrates the syntax of the getElementsByTagName():

```
let elements = document.getElementsByTagName(tagName);
```

```
let btn = document.getElementById('btnCount');

    btn.addEventListener('click', () => {

       let headings = document.getElementsByTagName('h2');

       alert(`The number of H2 tags: ${headings.length}`);

    });
```

Introduction to the getElementsByClassName() method

The getElementsByClassName() method returns an array-like of objects of the child elements with a specified class name.

Here's the syntax of the getElementsByClassName() method:

getElementsByClassName(names)

In this syntax:

- names parameter represents one or more class names to match. Multiple class names are separated by space.

The method returns undefined if no element with the class names is found.

Please note that the getElementsByClassName() method is available on both the document element and any other DOM elements.

When you call the getElementsByClassName() method on the document element, it will search the entire document and return the matched elements:

let elements = document.getElementsByClassName(names);

Introduction to JavaScript Events

An event is an action that the web browser can detect and respond to, like a mouse click or a page load.

For example, you might want to display an alert when a user clicks a button.

An event may have an event handler, a function that runs when the event occurs. An event handler, also known as an event listener, listens for the event and executes when it happens.

To define a function that will be executed when the button is clicked, you need to register an event handler using the addEventListener() method

Eg.

```
let btn = document.querySelector('#btn');

function handleClick() {
   alert('It was clicked!');
}

btn.addEventListener('click', handleClick);
```

OR

```
let btn = document.querySelector('#btn');

btn.addEventListener('click', function(event) {
   alert(event.type); // click
});
```

OnLoad Event

Window.addEventListener('load',function(event){

Console.log("event loaded");

}

- **dblclick**: Fired when a pointing device button is clicked twice on a single element.
- **mousedown**: Fired when a pointing device button is pressed on an element.
- **mouseup**: Fired when a pointing device button is released over an element.
- **mouseover**: Fired when a pointing device is moved onto an element.
- **mouseout**: Fired when a pointing device is moved off an element.
- **mousemove**: Fired when a pointing device is moved while it is over an element.

- **submit**: Fired when a form is submitted.
- **change**: Fired when the value of an element changes (for input, select, and textarea elements).
- **input**: Fired when the value of an element is changed.

- **load**: Fired when the whole page has loaded, including all dependent resources such as stylesheets and images.
- **unload**: Fired when the document or a child resource is being unloaded.
- **scroll**: Fired when the document view or an element has been scrolled.

1. **oninput Event**
    - Update a paragraph with the current value of a text input.
2. **onclick Event**
    - Change the color of a div when it is clicked.
3. **onload Event**
    - Display an alert when the page loads.
4. **onmouseover Event**
    - Change the text color of a paragraph when the mouse is over it.