What is Document Object Model (DOM)

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

The DOM represents a document as a tree of nodes. It provides API that allows you to add, remove, and modify parts of the document effectively.

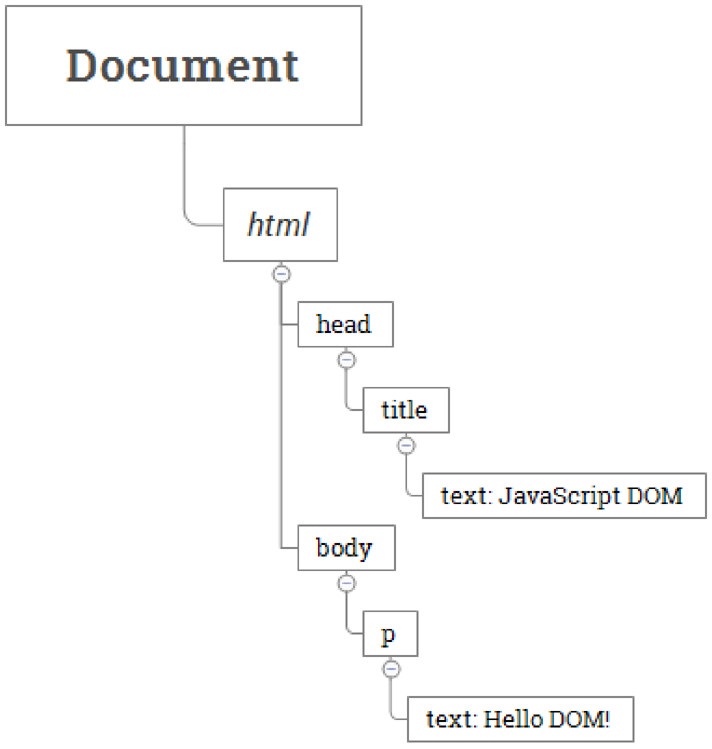
Note that the DOM is cross-platform and language-independent ways of manipulating HTML and XML documents.

A document as a hierarchy of nodes

The DOM represents an HTML or XML document as a hierarchy of nodes. Consider the following HTML document:

```html
<html>
    <head>
        <title>JavaScript DOM</title>
    </head>
    <body>
        <p>Hello DOM!</p>
    </body>
</html>
```

The following tree represents the above HTML document:



In this DOM tree, the document is the root node. The root node has one child 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 `<div>`.
`Node.TEXT_NODE`	3	The actual `Text` inside an `Element` or `Attr`.
`Node.CDATA_SECTION_NODE`	4	A `CDATASection`, such as `<!CDATA[[ … ]]>`.
`Node.PROCESSING_INSTRUCTION_NODE`	7	A `ProcessingInstruction` of an XML document, such as `<?xml-stylesheet … ?>`.
`Node.COMMENT_NODE`	8	A `Comment` node, such as `<!-- … -->`.
`Node.DOCUMENT_NODE`	9	A `Document` node.

Constant	Value	Description
Node.DOCUMENT_TYPE_NODE	10	A DocumentType node, such as <!DOCTYPE html>.
Node.DOCUMENT_FRAGMENT_NODE	11	A DocumentFragment node.

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

```
node.nodeType
```

You can compare the `nodeType` property with the above constants to determine the node type. For example:

```
if (node.nodeType == Node.ELEMENT_NODE) {
    // node is the element node
}
```

The nodeName and nodeValue properties

A node has two important properties: `nodeName` and `nodeValue` that provide specific information about the node.

The values of these properites depends on the node type. For example, if the node type is the element node, the `nodeName` is always the same as element's tag name and `nodeValue` is always `null`.

For this reason, it's better to test node type before using these properties:

```
if (node.nodeType == Node.ELEMENT_NODE) {
    let name = node.nodeName; // tag name like <p>
}
```

Node and Element

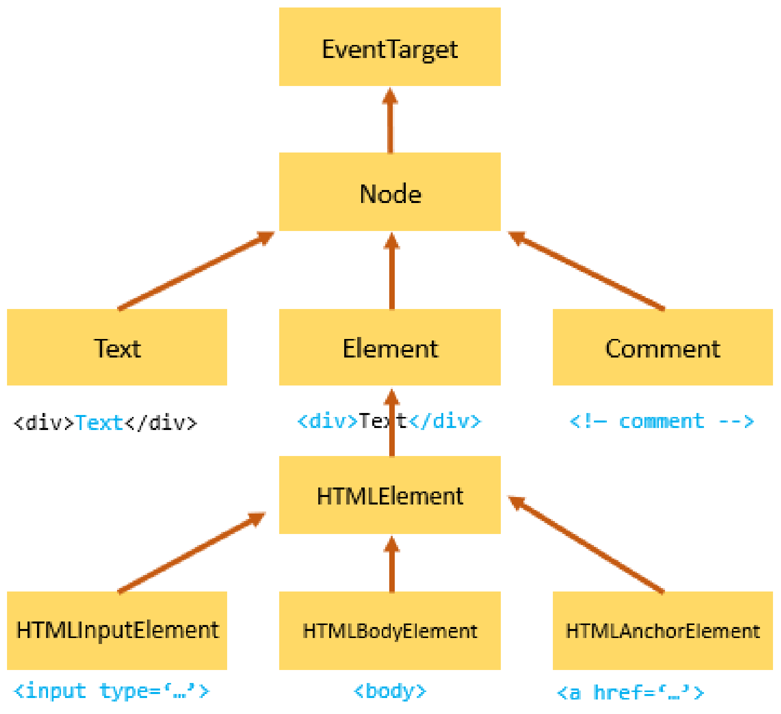
Sometime, it's easy to confuse between the `Node` and the `Element`.

A node is a generic name of any object in the DOM tree. It can be any built-in DOM element such as the document. Or it can be any HTML tag specified in the HTML document like `<div>` or `<p>`.

An element is a node with a specific node type `Node.ELEMENT_NODE`, which is equal to 1.

In other words, the node is generic type of the element. The element is a specific type of the node with the node type `Node.ELEMENT_NODE`.

The following picture illustrates the relationship between the `Node` and `Element` types:



Note that the `getElementById()` and `querySelector()` returns an object with the `Element` type while `getElementsByTagName()` or `querySelectorAll()` returns `NodeList` which is a collection of nodes.

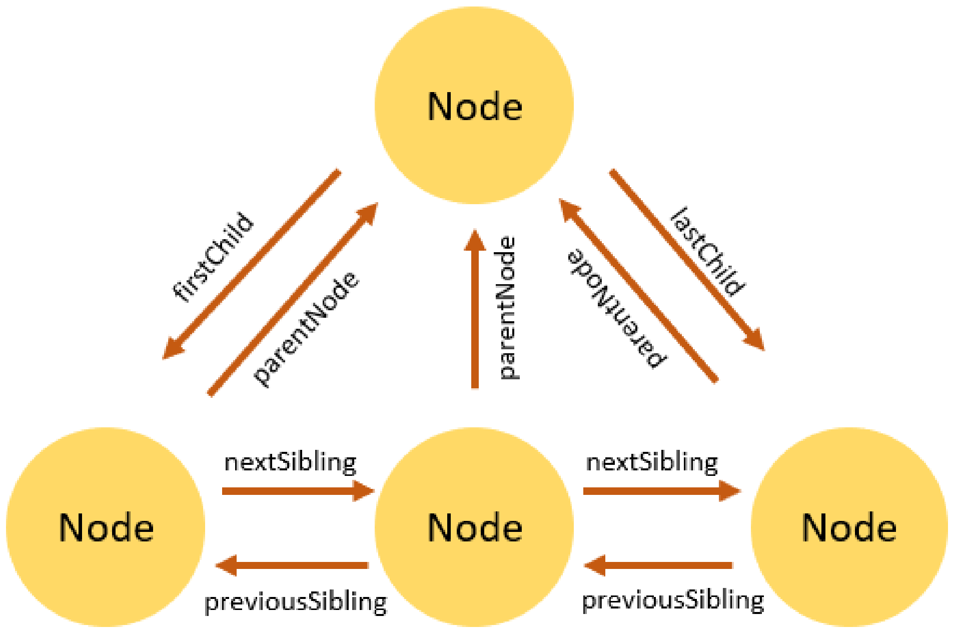
Node Relationships

Any node has relationships to other nodes in the DOM tree. The relationships are the same as the one described in a traditional family tree.

For example, `<body>` is a child nodeof the `<html>` node, and `<html>` is the parentof the `<body>` node.

The `<body>` node is the siblingof the `<head>` node because they share the same immediate parent, which is the `<html>` element.

The following picture illustrates the relationships between nodes:



Summary

An HTML or XML document can be represented as a tree of nodes, like a traditional family tree.

Each markup can be represented as a node with a specific node type.

Element is a specific type of node with the node type `Node.ELEMENT_NODE`.

In the DOM tree, a node has relationships with other nodes.