

The HTML DOM defines a standard for accessing and manipulating HTML documents.

What You Should Already Know

Before you continue you should have a basic understanding of the following:

- HTML / XHTML
- JavaScript

If you want to study these subjects first, find the tutorials on our [Home page](#).

What is the DOM?

The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents like HTML and XML:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The DOM is separated into 3 different parts / levels:

- Core DOM - standard model for any structured document
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

The DOM defines the **objects and properties** of all document elements, and the **methods** (interface) to access them.

What is the XML DOM?

The XML DOM defines the **objects and properties** of all XML elements, and the **methods** (interface) to access them.

If you want to study the XML DOM, find the XML DOM tutorial on our [Home page](#).

What is the HTML DOM?

The HTML DOM is:

- A standard object model for HTML
- A standard programming interface for HTML
- Platform- and language-independent
- A W3C standard

The HTML DOM defines the **objects and properties** of all HTML elements, and the **methods** (interface) to access them.

In other words: **The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**

DOM Nodes

According to the DOM, everything in an HTML document is a node.

The DOM says:

- The entire document is a document node
- Every HTML element is an element node
- The text in the HTML elements are text nodes
- Every HTML attribute is an attribute node
- Comments are comment nodes

DOM Example

Look at the following HTML document:

```
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>
```

The root node in the HTML above is `<html>`. All other nodes in the document are contained within `<html>`.

The `<html>` node has two child nodes; `<head>` and `<body>`.

The `<head>` node holds a `<title>` node. The `<body>` node holds a `<h1>` and `<p>` node.

Text is Always Stored in Text Nodes

A common error in DOM processing is to expect an element node to contain text.

However, the text of an element node is stored in a text node.

In this example: `<title>DOM Tutorial</title>`, the element node `<title>`, holds a text node with the value "DOM Tutorial".

"DOM Tutorial" is **not** the value of the `<title>` element!

However, in the HTML DOM the value of the text node can be accessed by the **innerHTML** property.

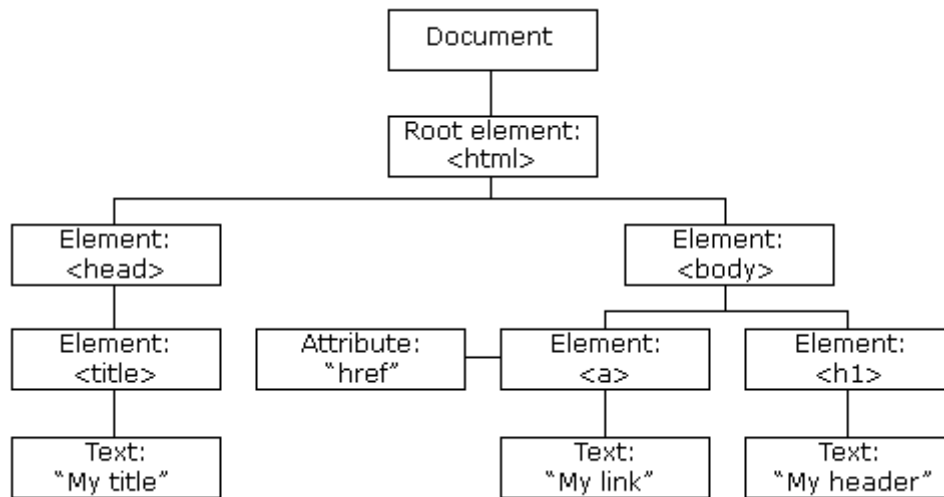
You can read more about the innerHTML property in a later chapter.

The HTML DOM Node Tree

The HTML DOM views a HTML document as a tree-structure. The tree structure is called a **node-tree**.

All nodes can be accessed through the tree. Their contents can be modified or deleted, and new elements can be created.

The node tree below shows the set of nodes, and the connections between them. The tree starts at the root node and branches out to the text nodes at the lowest level of the tree:



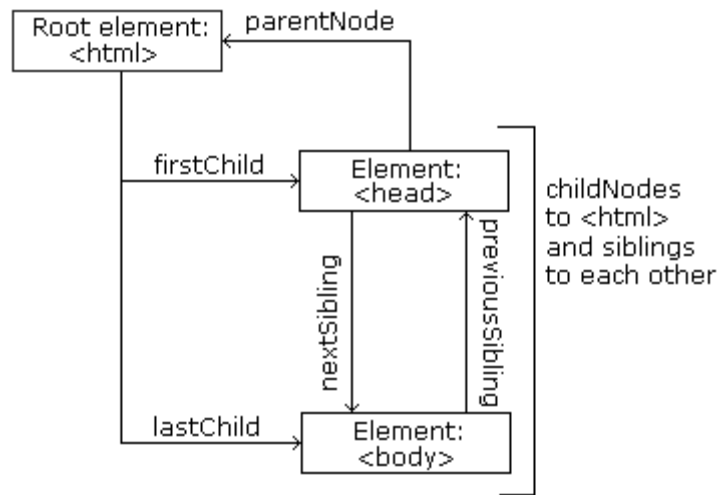
Node Parents, Children, and Siblings

The nodes in the node tree have a hierarchical relationship to each other.

The terms parent, child, and sibling are used to describe the relationships. Parent nodes have children. Children on the same level are called siblings (brothers or sisters).

- In a node tree, the top node is called the root
- Every node, except the root, has exactly one parent node
- A node can have any number of children
- A leaf is a node with no children
- Siblings are nodes with the same parent

The following image illustrates a part of the node tree and the relationship between the nodes:



Look at the following HTML fragment:

```

<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>

```

From the HTML above:

- The <html> node has no parent node; it is the root node
- The parent node of the <head> and <body> nodes is the <html> node
- The parent node of the "Hello world!" text node is the <p> node

and:

- The <html> node has two child nodes; <head> and <body>
- The <head> node has one child node; the <title> node
- The <title> node also has one child node; the text node "DOM Tutorial"
- The <h1> and <p> nodes are siblings, and both child nodes of <body>

First Child - Last Child

From the HTML above:

- the <head> element is the first child of the <html> element, and the <body> element is the last child of the <html> element
- the <h1> element is the first child of the <body> element, and the <p> element is the last child of the <body> element

Programming Interface

In the DOM, HTML documents consist of a set of node objects. The nodes can be accessed with JavaScript or other programming languages. In this tutorial we will use JavaScript.

The programming interface of the DOM is defined by standard properties and methods.

Properties are often referred to as something that is (i.e. the name of a node).

Methods are often referred to as something that is done (i.e. remove a node).

HTML DOM Properties

Some DOM properties:

- `x.innerHTML` - the text value of `x`
- `x.nodeName` - the name of `x`
- `x.nodeValue` - the value of `x`
- `x.parentNode` - the parent node of `x`
- `x.childNodes` - the child nodes of `x`
- `x.attributes` - the attributes nodes of `x`

Note: In the list above, `x` is a node object (HTML element).

HTML DOM Methods

Some DOM methods:

- `x.getElementById(id)` - get the element with a specified `id`
- `x.getElementsByTagName(name)` - get all elements with a specified tag name
- `x.appendChild(node)` - insert a child node to `x`
- `x.removeChild(node)` - remove a child node from `x`

Note: In the list above, `x` is a node object (HTML element).

The innerHTML Property

The easiest way to get or modify the content of an element is by using the `innerHTML` property.

`innerHTML` is not a part of the W3C DOM specification. However, it is supported by all major browsers.

The `innerHTML` property is useful for returning or replacing the content of HTML elements (including `<html>` and `<body>`), it can also be used to view the source of a page that has been dynamically modified.

Example

The following code to gets the `innerHTML` (text) from the `<p>` element with `id="intro"`:

Example

```
<html>
<body>
```

```
<p id="intro">Hello World!</p>

<script type="text/javascript">
txt=document.getElementById("intro").innerHTML;
document.write("<p>The text from the intro paragraph: " + txt +
"</p>");
</script>

</body>
</html>
```

childNodes and nodeValue

We can also use the `childNodes` and `nodeValue` properties to get the content of an element.

The following code to gets the value of the `<p>` element with `id="intro"`:

Example

```
<html>
<body>

<p id="intro">Hello World!</p>

<script type="text/javascript">
txt=document.getElementById("intro").childNodes[0].nodeValue;
document.write("<p>The text from the intro paragraph: " + txt +
"</p>");
</script>

</body>
</html>
```

[Try it yourself >](#)

In the example above, `getElementById` is a method, while `childNodes` and `nodeValue` are properties.

In this tutorial we will mostly use the `innerHTML` property. However, learning the method above is useful for understanding the tree structure of the DOM and dealing with XML files.

Accessing Nodes

You can access a node in three ways:

1. By using the `getElementById()` method
2. By using the `getElementsByName()` method
3. By navigating the node tree, using the node relationships

The `getElementById()` Method

The `getElementById()` method returns the element with the specified ID:

Syntax

```
node.getElementById( "id" );
```

The following example gets the element with id="intro":

Example

```
document.getElementById( "intro" );
```

[Try it yourself »](#)

Note: The getElementById() method doesn't work in XML.

The getElementsByTagName() Method

getElementsByTagName() returns all elements with a specified tag name.

Syntax

```
node.getElementsByTagName( "tagname" );
```

The following example returns a nodeList of all <p> elements in the document:

Example 1

```
document.getElementsByTagName( "p" );
```

[Try it yourself »](#)

The following example returns a nodeList of all <p> elements that are descendants of the element with id="main":

Example 2

```
document.getElementById( 'main' ).getElementsByTagName( "p" );
```

[Try it yourself »](#)

DOM Node List

The `getElementsByName()` method returns a node-list. A node-list is an array of nodes.

The following code selects all `<p>` nodes in a node-list:

Example

```
x=document.getElementsByTagName("p");
```

The nodes can be accessed by index number. To access the second `<p>` you can write:

```
y=x[1];
```

[Try it yourself »](#)

Note: The index starts at 0.

You will learn more about node-lists in a later chapter of this tutorial.

DOM Node List Length

The `length` property defines the number of nodes in a node-list.

You can loop through a node-list by using the `length` property:

Example

```
x=document.getElementsByTagName("p");

for (i=0;i<x.length;i++)
{
document.write(x[i].innerHTML);
document.write("<br />");
}
```

[Try it yourself »](#)

Example explained:

1. Get all `<p>` element nodes
 2. For each `<p>` element, output the value of its text node
-

Navigating Node Relationships

The three properties; parentNode, firstChild, and lastChild, follow the document structure and allow short-distance travel in a document.

Look at the following HTML fragment:

```
<html>
<body>

<p>Hello World!</p>
<div>
  <p>The DOM is very useful!</p>
  <p>This example demonstrates node relationships.</p>
</div>

</body>
</html>
```

In the HTML code above, the first p element is the first child node (firstChild) of the body element, and the div element is the last child node (lastChild) of the body element. The parent node (parentNode) of the first p element and the div element, is the the body element, and the parent node of the p elements inside the div element, is the div element.

The firstChild property can also be used to access the text of an element:

Example

```
<html>
<body>

<p id="intro">Hello World!</p>

<script type="text/javascript">
x=document.getElementById("intro");
document.write(x.firstChild.nodeValue);
</script>

</body>
</html>
```

[Try it yourself »](#)

DOM Root Nodes

There are two special document properties that allow access to the tags:

- document.documentElement - returns the root node of the document
- document.body - gives direct access to the <body> tag

Events

Every element on a web page has certain events which can trigger JavaScript functions. For example, we can use the `onClick` event of a button element to indicate that a function will run when a user clicks on the button. We define the events in the HTML elements.

Examples of events:

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input box in an HTML form
- Submitting an HTML form
- A keystroke

Note: Events are normally used in combination with functions, and the function will not be executed before the event occurs!

onload and onUnload

The `onload` and `onUnload` events are triggered when the user enters or leaves a page.

The `onload` event is often used to check the visitor's browser type and version, and load the proper version of the web page based on that information.

Both the `onload` and `onUnload` events are also often used to deal with cookies that should be set when a user enters or leaves a page. For example, you could have a popup asking for the user's name upon his first arrival to your page. The name is then stored in a cookie. Next time the visitor arrives at your page, you could have another popup saying something like: "Welcome John Doe!".

onFocus, onBlur and onChange

The `onFocus`, `onBlur` and `onChange` events are often used in combination with validation of form fields.

Below is an example of how to use an `onChange` event. The `checkEmail()` function will be called whenever the user changes the content of the e-mail field:

```
E-mail: <input type="text" id="email" onchange="checkEmail()" />
```
