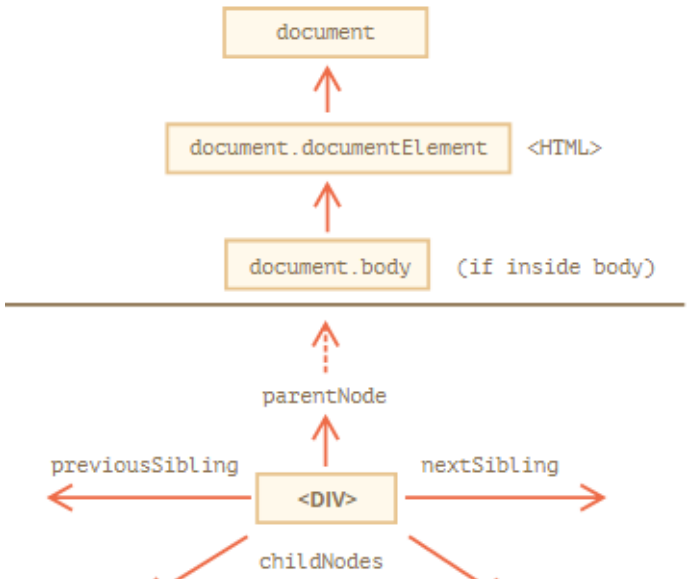
# Children, Parent & Traversing the DOM | JavaScript Tutorial In Hindi #15

In the previous tutorial, we have learned how to use the built-in methods of the document object to access HTML elements by ID, class, tag name, and query selectors. Often, we want to move through the DOM without specifying every element beforehand. In this tutorial, we will go over how to traverse the DOM using parent, child, and sibling properties. As we know that all the operations on the DOM start with the document object. It is the main entry point, and we can access any node from it. Here is a diagram of links that allow for travel between DOM nodes:




We can traverse the DOM in three directions, downwards, upwards and sideways. Each type of traversal uses a different method.

**Traversing downwards:-**

There are two methods to traverse downwards, the first one is querySelector and the second one is children.

**querySelector or querySelectorAll:-**

To traverse downwards from a specific element, we can use querySelector( ) or querySelectorAll( ). The **querySelector()** returns the first element within the document that matches the specified selector whereas the **querySelectorAll()** returns the NodeList representing a list of the document's elements that match the specified group of selectors.

```
<div class="add">
<h2 class="add__title">title</h2>
</div>
const component = document.querySelector('.add')
console.log(component)
```

```
> console.log(component)
    ▶ <div class="add">…</div>
< undefined
```

**children:-**

The property that lets you select direct descendants is called children . It selects elements that are immediately nested in another element. The children returns a HTML Collection that updates when children elements are changed.

```
const items= document.querySelector('.myclass')
const l_Items = items.children
console.log(l_Items)
```

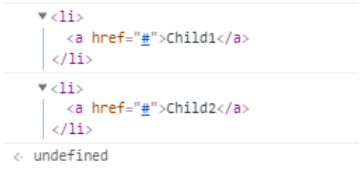
```
▼ HTMLCollection(5) [li, li, li, li, li]
  ▶ 0: li
  ▶ 1: li
  ▶ 2: li
  ▶ 3: li
  ▶ 4: li
    length: 5
  ▶ __proto__: HTMLCollection
```

To return the first and last child of a node, use the ***firstChild*** and ***lastChild*** property. The node can be of any node type, including text node, comment node, and element node. Similarly, ***firstElementChild*** and ***lastElementChild*** return the first and last child Element node, and the ***childNodes*** returns a live NodeList of all child nodes of any node type of a specified node.

**Selecting a specific child:-**

While traversing the DOM, we can select the nth-item in the list from both NodeLists and HTML Collections. For this, we use the index of the element. Similarly, we do in the case of the array to select a specific element.

```
const mylist = document.querySelectorAll('li')
const firstItem = mylist[0]
const secondItem = mylist[1]
console.log(firstItem)
console.log(secondItem)
```



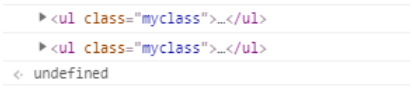

**Traversing upwards:-**

There is one method to traverse upwards: parentElement

**parentElement:-**

The property that let us select the parent element is known as parentElement. The parentElement returns *null* if the parent node is not an element node. Following is the example

```
const mylist = document.querySelectorAll('li')
const firstItem = mylist[0]
const secondItem = mylist[1]
console.log(firstItem.parentElement)
console.log(secondItem.parentElement)
```



**Traversing sideways:-**

There are two methods to traverse sideways. One of them is nextElementSibling, and the other one is previousElementSibling.

**nextElementSibling:-**

To select the next element, we use the nextElementSibling. The difference between this property and nextSibling is that nextSibling returns the next sibling node as an element node, a text node or a comment node, while nextElementSibling returns the next sibling node as an element node and ignores the text and comment nodes.

```
const item1 = document.querySelector('li')
const item2 = item1.nextElementSibling
console.log(item2)
```


**previousElementSibling:-**

To select the previous element, we use previousElementSibling. The difference between this property and previousSibling, is that previousSibling returns the previous sibling node as an element node, a text node or a comment node, while previousElementSibling returns the previous sibling node as an element node and ignores the text and comment nodes.

```
const item5 = document.querySelectorAll('li')[1]
const item6 = item5.previousElementSibling
console.log(item6)
```


**Node Type:-**

The **nodeType** property is an integer that identifies what the node is. It differentiate between different kind of nodes from each other, such as elements, text and comments. The syntax is:

```
var type = node.nodeType;
```

It will return an integer which specifies the type of the node.

| Constant | Value | Description |
|---|---|---|
| Node.ELEMENT_NODE | 1 | An Element node like <h1> or <p>. |
| Node.ATTRIBUTE_NODE | 2 | An Attribute of an Element. |
| Node.TEXT_NODE | 3 | The actual Text inside an Element |
| Node.COMMENT_NODE | 8 | A Comment node |
| Node.DOCUMENT_NODE | 9 | A Document node. |
| Node.DOCUMENT_TYPE_NODE | 10 | A DocumentType(<!DOCTYPE html>) node |
| Node.DOCUMENT_FRAGMENT_NODE | 11 | A DocumentFragment node. |

**Website.html code as described/written in the video**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <div class="container">
        <h1 id="heading"> Welcome to Code With Harry</h1>
        <div id="myfirst" class="child red good" id="first">child 1

            <ul class="this">
                <li class="childul">this</li>
                <li class="childul">is</li>
                <li class="childul">a</li>
                <li class="childul">list </li>
                <li class="childul">of my dreams</li>
```

```
            </ul>
        </div>
        <div class="child">child 2</div>
        <div class="child red">child 3</div>
        <div class="child">child 4</div>
        <form action="none.html" method="post">
            <a href="//codewithharry.com">Go to Code With Harry</a>
            <br>
            <br>
            Search this website: <input type="text" name="Hello" id="">
            <input type="button" value="submit">
        </form>
    </div>
    <br>
    <div class="no">this is a dummy div1</div>
    <div class="no">this is a dummy div2</div>
    <div class="no">this is a dummy div3</div>
</body>
<!-- <script src="js/tut12.js"></script> -->
<!-- <script src="js/tut14.js"></script> -->
<script src="js/tut15.js"></script>
</html>
```

**JavaScript code as described/written in the video**

```
console.log('Welcome to tutorial 15');

let cont = document.querySelector('.no');
cont = document.querySelector('.container');
let nodeName = cont.childNodes[1].nodeName;
let nodeType = cont.childNodes[1].nodeType;
// console.log(nodeName)
// console.log(nodeType)
// Node types
// 1. Element
// 2. Attribute
// 3. Text Node
// 8. Comment
```

```
// 9. document
// 10. docType

// console.log(cont.childNodes);
// console.log(cont.children);

let container = document.querySelector('div.container');

// console.log(container.children[1].children[0].children);

// console.log(container.firstChild);
// console.log(container.firstElementChild);

// console.log(container.lastChild);
// console.log(container.lastElementChild);
// console.log(container.children);
// console.log(container.childElementCount); // Count of child elements

console.log(container.firstElementChild.parentNode);
console.log(container.firstElementChild.nextSibling);
console.log(container.firstElementChild.nextElementSibling);
console.log(container.firstElementChild.nextElementSibling.nextElementSibling);
```

Previous Next

CodeWithHarry