Tutorial : More DOM Modification

More DOM modification

Accessing nodes

The JavaScript object document.documentElement is the root of the DOM tree. Ex: `let html = document.documentElement;` assigns the `html` variable with the HTML document's root node.

DOM nodes have properties for accessing a node's parent, children, and siblings:

1. The parentNode property refers to the node's parent. Ex: In the figure below, the ol node is the `parentNode` for all li nodes.
2. The childNodes property is an array-like collection of objects for each of the node's children. Ex: In the figure below, the li nodes and whitespace text nodes are the ol node's `childNodes`.
3. The children property is similar to the `childNodes` except the array contains only element nodes and no text nodes. Ex: In the figure below, the li nodes are the ol node's `children`.
4. The nextElementSibling property refers to the element node with the same parent following the current node in the document. Ex: In the figure below, the ol node is the p node's `nextElementSibling`.
5. The previousElementSibling property refers to the element node with the same parent preceding the current node in the document. Ex: In the figure below, the p node is the ol node's `previousElementSibling`.

A common error is to use the `childNodes` property instead of the `children` property when iterating through the items of a list. The `children` property only contains the list items, while the `childNodes` property also contains the whitespace text nodes between the list items.

Figure Example HTML for node properties.

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8">
      <title>Geologic eons of earth</title>
   </head>
   <body>
      <p>The four geologic eons on earth:</p>
      <ol>
         <li><a href="https://en.wikipedia.org/wiki/Hadean">Hadean</a></li>
         <li><a href="https://en.wikipedia.org/wiki/Archean">Archean</a></li>
         <li><a href="https://en.wikipedia.org/wiki/Proterozoic">Proterozoic</a></li>
         <li><a href="https://en.wikipedia.org/wiki/Phanerozoic">Phanerozoic</a></li>
      </ol>
```

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

Modifying the DOM structure

Various DOM node methods can change a node's location within the DOM or remove nodes:

- The appendChild() method appends a DOM node to the object's child nodes. The code below moves the ordered list's first list item to the last list item of the same ordered list.

```
ol = document.getElementsByTagName("ol")[0];
li = ol.getElementsByTagName("li")[0];
ol.appendChild(li);
```

- The insertBefore() method inserts a DOM node as a child node before an object's existing child node. The code below moves the ordered list's first list item before the fourth list item.

```
ol = document.getElementsByTagName("ol")[0];
items = ol.getElementsByTagName("li");
ol.insertBefore(items[0], items[3]);
```

- The removeChild() method removes a node from the object's children. The most common usage pattern is to get a DOM node, n, and call `removeChild()` on n's parent passing n as an argument. Ex: `n.parentNode.removeChild(n)`

The following methods are for creating new nodes or duplicating existing nodes:

- The `document` method createElement() returns a new element node, created from a string argument that names the HTML element. Ex: `document.createElement("p")` creates a new paragraph node.
- The `document` method createTextNode() returns a new text node containing the text specified by a string argument. Ex: `document.createTextNode("Hello there!")` creates the text node with "Hello there!", which can then be appended to an element node.
- The node method cloneNode() returns an identical copy of a DOM node. The method's boolean argument indicates whether the method should also clone the node's children. Ex: `x.cloneNode(true)` creates an identical tree rooted at x, but `x.cloneNode(false)` creates only a single node identical to x. A common error is to forget to modify any id attributes in the cloned tree. The `cloneNode()` method does not ensure that new nodes have unique id attributes.

After creating or cloning a node, the node does not appear in the webpage until the node is

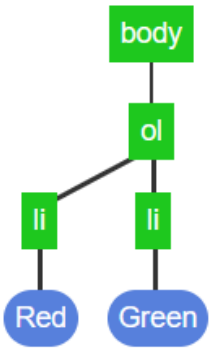
attached to the DOM tree. A programmer must use `appendChild()` or `insertBefore()` to add the new node to the existing DOM tree.

Example :

1. The HTML file defines an ordered list with two colors.

example.html

```
<!DOCTYPE HTML>
<html lang="en">
   <title>DOM Example</title>
   <script src="script.js" defer></script>
   <body>
      <ol>
         <li>Red</li>
         <li>Green</li>
      </ol>
   </body>
</html>
```



2. script.js

```
let listNode = document.createElement("li");
let textNode = document.createTextNode("Blue");
listNode.appendChild(textNode);

let colorList = document.querySelector("ol");
colorList.appendChild(listNode);

let itemNodes = colorList.querySelectorAll("li");
let clonedNode = itemNodes[0].cloneNode(true);
colorList.insertBefore(clonedNode, itemNodes[2]);
```

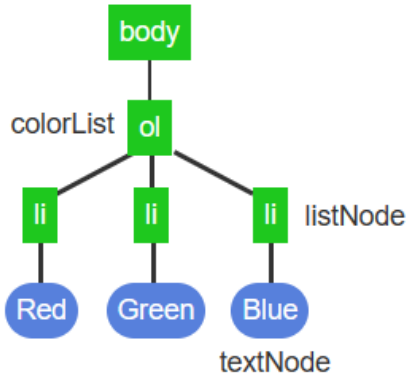
1. Red
2. Green

The HTML file defines an ordered list with two colors.

3. After creating an element node and a text node, the text node is appended to the element node. listNode becomes textNode's parent.

example.html

```
<!DOCTYPE HTML>
<html lang="en">
   <title>DOM Example</title>
   <script src="script.js" defer></script>
   <body>
      <ol>
         <li>Red</li>
         <li>Green</li>
      </ol>
   </body>
</html>
```



script.js

```
let listNode = document.createElement("li");
let textNode = document.createTextNode("Blue");
listNode.appendChild(textNode);

let colorList = document.querySelector("ol");
colorList.appendChild(listNode);

let itemNodes = colorList.querySelectorAll("li");
let clonedNode = itemNodes[0].cloneNode(true);
colorList.insertBefore(clonedNode, itemNodes[2]);
```

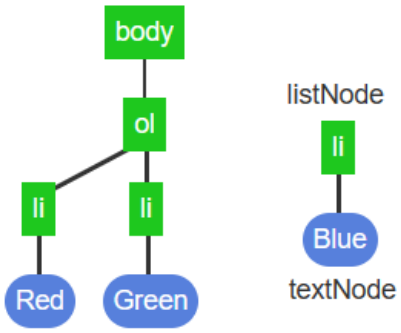
1. Red
2. Green
3. Blue

appendChild() appends listNode to colorList's child nodes.
Since listNode is now attached to the DOM, the "Blue" list item appears in the browser.

4.The cloneNode() method creates a copy of itemNodes[0] and child, duplicating the "Red" list item.

example.html

```
<!DOCTYPE HTML>
<html lang="en">
   <title>DOM Example</title>
   <script src="script.js" defer></script>
   <body>
      <ol>
         <li>Red</li>
         <li>Green</li>
      </ol>
   </body>
</html>
```



script.js

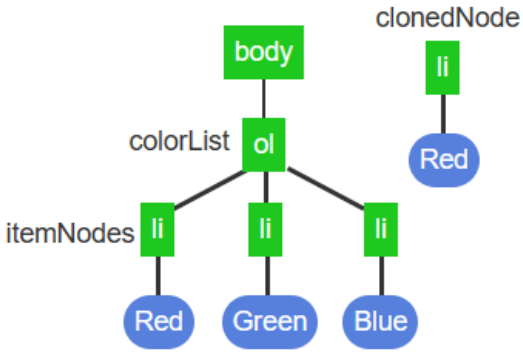
```
let listNode = document.createElement("li");
let textNode = document.createTextNode("Blue");
listNode.appendChild(textNode);

let colorList = document.querySelector("ol");
colorList.appendChild(listNode);

let itemNodes = colorList.querySelectorAll("li");
let clonedNode = itemNodes[0].cloneNode(true);
colorList.insertBefore(clonedNode, itemNodes[2]);
```

1. Red
2. Green

After creating an element node and a text node, the text node is appended to the element node. listNode becomes textNode's parent.

example.html

```
<!DOCTYPE HTML>
<html lang="en">
   <title>DOM Example</title>
   <script src="script.js" defer></script>
   <body>
      <ol>
         <li>Red</li>
         <li>Green</li>
      </ol>
   </body>
</html>
```



script.js

```
let listNode = document.createElement("li");
let textNode = document.createTextNode("Blue");
listNode.appendChild(textNode);

let colorList = document.querySelector("ol");
colorList.appendChild(listNode);

let itemNodes = colorList.querySelectorAll("li");
let clonedNode = itemNodes[0].cloneNode(true);
colorList.insertBefore(clonedNode, itemNodes[2]);
```

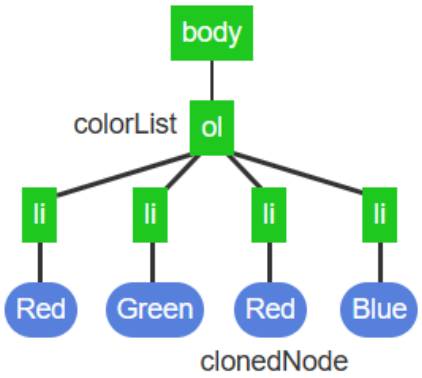
1. Red
2. Green
3. Blue

The cloneNode() method creates a copy of itemNodes[0] and child, duplicating the "Red" list item.

5.insertBefore() inserts the cloned "Red" list item before the "Blue" list item, which changes the DOM.

example.html

```
<!DOCTYPE HTML>
<html lang="en">
   <title>DOM Example</title>
   <script src="script.js" defer></script>
   <body>
      <ol>
         <li>Red</li>
         <li>Green</li>
      </ol>
   </body>
</html>
```



script.js

```
let listNode = document.createElement("li");
let textNode = document.createTextNode("Blue");
listNode.appendChild(textNode);

let colorList = document.querySelector("ol");
colorList.appendChild(listNode);

let itemNodes = colorList.querySelectorAll("li");
let clonedNode = itemNodes[0].cloneNode(true);
colorList.insertBefore(clonedNode, itemNodes[2]);
```

1. Red
2. Green
3. Red
4. Blue

insertBefore() inserts the cloned "Red" list item before the "Blue" list item, which changes the DOM.

Checkpoint :
Adding and removing DOM Nodes :

insertBefore()	A method on a DOM node which moves one DOM node to be a previous sibling to another DOM node. The code `parent.insertBefore(nodeToMove, existingNode)` causes nodeToMove to be a child of parent immediately preceding existingNode.
removeChild()	A method on a DOM node that deletes a DOM node from the DOM tree. The code `parent.removeChild(nodeToRemove)` causes nodeToRemove to be deleted.
appendChild()	A method on a DOM node that turns another DOM node into the first DOM node's last child. The code `parent.appendChild(nodeToMove)` causes nodeToMove to become parent's last child.