Implementation of Tree in Document Object Model (DOM)

Kevin Fernaldy 13516109
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13516109@std.stei.itb.ac.id

***Abstract*—Tree is a basic component of Informatics Engineering. From representing data structure to mapping an Artificial Intelligence decision, tree has been a useful, yet simple tool. In web design, tree is a key component in designing a web page. Tree is also a major help in understanding contents of a certain document in web application. How can a tree help in web design?**

***Keywords*—Document Object Model, Tree, HTML, XML.**

I. Introduction

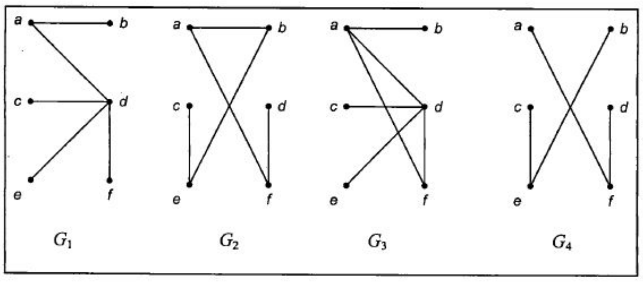
In web design, we might have known two file types called HTML and XML. HTML or Hypertext Markup Language is a most common file in representing a web page. Whereas XML or Extensible Markup Language is a simple, very flexible text format that plays an important role in exchanging a wide variety of data on the web and elsewhere [1]. These two files are the most common files used in web designing.

Often, we need a logical view of HTML and XML documents, each with its reasons. A logical view of HTML is needed in changing the contents of the HTML file dynamically. In XML, a logical view is needed in understanding the contents of the XML file. This logical view is called DOM or Document Object Model.

This paper will explore a simple explanation about DOM, the relation between tree and DOM, and the analysis of the relation.

II. Tree

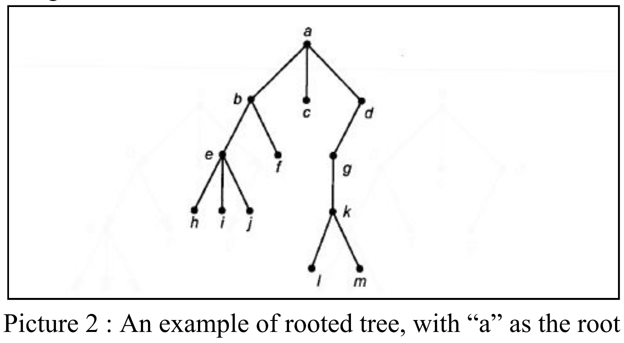
Tree is a special case of graph. A tree is a connected, undirected graph, in which does not contain a circuit [2].



Picture 1 : G1 and G2 is a tree, whereas G3 and G4 is not.
Source : R. Munir, "Matematika Diskrit Revisi Keempat", Bandung, Indonesia: Informatika Bandung, 2010, pp. 444.

A tree can be a free tree, and a rooted tree. A rooted tree is a tree in which a vertex is treated as a "root", and the vertex connected to the root are drawn away from the vertex, treated as a "leaf" or "branch"of the parent, depending on the case [3]. A free tree doesn't have this parent-child representation and is treated like a normal graph.

A root has zero entry-degree into its vertex, whereas all of the other vertices has one entry-degree. A vertex that has zero exit-degree is called a leaf, whereas a vertex that has non-zero exit degree is called a branch.



Picture 2 : An example of rooted tree, with "a" as the root
Source : R. Munir, "Matematika Diskrit Revisi Keempat", Bandung, Indonesia: Informatika Bandung, 2010, pp. 459.

III. Document Object Model (DOM)

According to Peter-Paul Koch, a DOM is a model of elements in a certain document that represents how each elements are related to each other and to the topmost structure, i.e. the document itself [4]. DOM was invented as a standard and a requirement of JavaScript. Because JavaScript gives the ability to change things in a web page from user-generated events, an access to HTML element is needed to change the HTML structure in response of the user events. This is where DOM is invented in order to give a complete access of the HTML elements to the JavaScript.

DOM is divided into 5 levels, ranging from Level 0 to Level 4. DOM Level 0 DOM was invented by Netscape [5], with a very simple access to the elements in HTML. Level 0 DOM was then implemented into Netscape 2 and Netscape 3. After seeing the Netscape inventing the Level 0 DOM, Microsoft follows the steps of Netscape and implement Level 0 DOM into their Explorer 3.0. However, because of no standards were made in implementing Level 0 DOM, there was a difference with Netscape and Microsoft implementations. Netscape Level 0 DOM can access forms, links, and images. Microsoft can access all of them too, except for images. This creates a need for compatibility checking for web developers in using the Level 0 DOM for Netscape and Explorer.

Level 1 DOM was invented by the World Wide Web Consortium (W3C). W3C reinforces the standard of DOM for compatibility across all web browsers. W3C also adds a new feature of accessing every elements of an XML and HTML documents. This version of DOM were adopted by numerous web browser development team, such as Microsoft, Mozilla, and Opera.

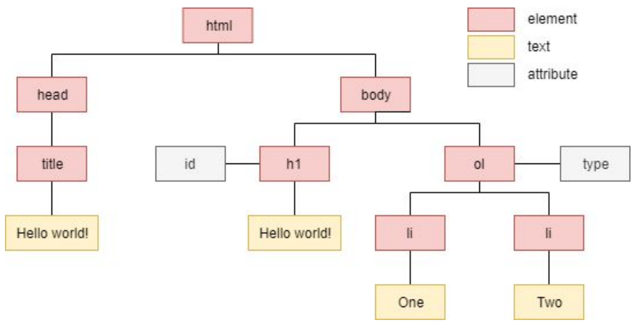
IV. Relation of Tree and DOM

To represent a DOM in a tree, we will be using a rooted tree. The first tag of HTML and XML documents will be the parents of all nodes, or the root element.

Suppose we have a HTML file called sample.html below.

```
<html>
  <head>
    <title>Hello world!</title>
  </head>
  <body>
    <h1 id="hello">Hello world!</h1>
    <ol type="I">
      <li>One</li>
      <li>Two</li>
    </ol>
  </body>
</html>
```

Every tags of HTML is treated as an element node in DOM, with a special case of <html> tag is a root element. The attributes inside the HTML tag is treated as attribute node in DOM. As for the text inside the HTML tags, it is treated as the text node in the DOM. By using a tree, we can represent the DOM of the HTML above like this.



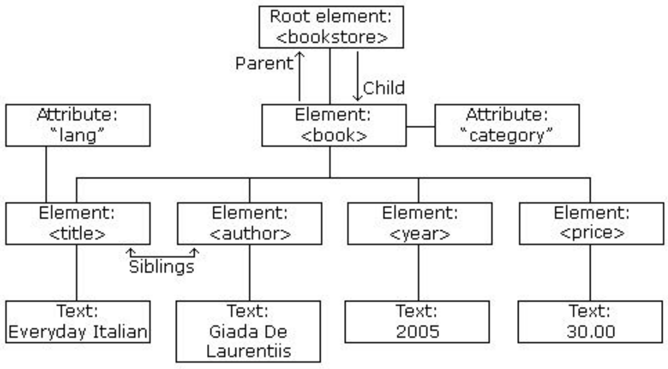
Picture 3 : DOM tree representation of sample.html

From the picture 3, all of the elements with their attributes and text are shown clearly with the tree. When we want to access and change the DOM of HTML, we can use this tree representation to help us in doing so.

As for a XML file, suppose we have a XML file called books.xml [6] below.

```
<book category="children">
<title lang="en">Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>29.99</price>
</book>
<book category="web">
<title lang="en">XQuery Kick Start</title>
<author>James McGovern</author>
<author>Per Bothner</author>
<author>Kurt Cagle</author>
<author>James Linn</author>
<author>Vaidyanathan Nagarajan</author>
<year>2003</year>
<price>49.99</price>
</book>
<book category="web" cover="paperback">
<title lang="en">Learning XML</title>
<author>Erik T. Ray</author>
<year>2003</year>
<price>39.95</price>
</book>
</bookstore>
```

As in the HTML, all elements in the XML are treated as an element node in DOM, with the first element is treated as a root element. The attributes of the XML elements are treated as an attribute node of the corresponding element. The value or data of an element is treated as the text node. as By using a tree, we can represent the DOM of books.xml as below



Picture 4 : DOM tree representation of books.xml
Source : https://www.w3schools.com/xml/nodetree.gif

IV. Analysis of The DOM Tree

From the DOM trees in the Picture 3 and 4, there are 3 categories of nodes.

1. Element node
   Element node is a representation of HTML and XML tags. Tags are the words that is contained in the <> symbol. In HTML, this symbol represents an element with certain function in the web page, such as <h1> for writing a header, <input> for creating an input field, and <br> for creating a break line. In XML, the <> symbol represents the information of the data that is contained in the text node below the tag.
2. Text node
   Text node is a representation of text from the document. Tags and text are different. Text is not contained in the <> symbol. Text are the words that will be rendered into the web page
3. Attribute node
   Attribute node is an representation of information of elements. Each element in HTML and XML may have multiple attributes that has its each function. This attributes are represented as attribute node. This attribute node are the information about the parent element node.

In the Picture 3 and 4, there are 2 relations between nodes. These relations determines how a DOM is rendered as a web page, and how to access each of the node in the DOM Tree.

1. Parent-child relation
   Parent-child relation is the most simple relation in the DOM tree. A parent is a node which has node(s) connected under it, whereas a child is a node which has a connection to a node above it. In the DOM tree, the root element is the parent of all elements in the document. Parent-child relations are easy to spot, for example in the Picture 3,
   - element <ol> is a parent to all of the <li> elements. Each of the element <li> is a child to the element <ol>
   - attribute "id" and text "Hello world!" is the children of element <h1>
   - element <h1> and element <ol> element is not parent-child element, as they are connected to the same parent, which is element <body>
2. Siblings relation
   Siblings relation is a relation between the children with the same parents. All of the children in the DOM tree that has the same parent has siblings relation. For example, in the Picture 4,
   - element <author> and <year> has a siblings relation, since they have a same parent node, element <book>
   - element <book> has no siblings relation, because it is the only child of parent <bookstore>

From the relations above, we can get an idea of how to access every single nodes in the DOM. There are 2 ways of accessing a node in the DOM tree.

1. Accessing from the root element
   Accessing from the root element is the most primitive way to access a certain node. We have to count all of the child nodes, selecting the index of the child node that contains the node we wan't to access, and doing it again and again until we reach the desired node. For example, we want to reach the text node "Hello world!" in the element <title>. With the JavaScript syntax, we can reach it from the root element with this command.

```
document.childNodes[0].childNodes[0].childNodes[0].c
hildNodes[0].data
```

   This method is not efficient, because we need to iterate the childNodes until we reach the desired node. This will get worse if the DOM tree depth is large.
2. Accessing by element's identifier
   Accessing by element's identifier is the most efficient way to access a node. There are many identifiers for an element inside a DOM. For HTML, there are id, class, and name. For XML, the elements are the identifier to the data. For example, in Picture 4, if we want to get the XQuery Kick Start book title, we simply only need to invoke this command

```
xmlDoc.getElementsByTagName("title")[1]
```

   The getElementsByTagName returns an array of data which contains the title of all the books inside the XML. By adding [1] at the end, we choose the data inside index 2 of the array, which is the XQuery Kick Start.

V. Benefits of Tree Implementation In DOM

Tree implementation in DOM provides benefits in web design. Some of them are :

1. Providing a clear and simple structure of the document, and relations between elements
   A raw HTML and XML file can be read and understood if the files are short and only contains simple data. Nowadays, HTML files may contain hundreds of elements and hundreds of attributes in order to create a good web page. XML files also contain thousand of dataset that will be a pain to read and analyze. By representing the file into a DOM tree, we can understand the structure of the HTML and XML files more clearly. Searching a node based on their attributes is as simple as searching the attribute nodes in the tree.
2. Providing an easier and more efficient methods in

modifying and accessing the DOM
Many JavaScript frameworks, such as AngularJS views the DOM of a web page as a tree structure. By viewing a DOM as a tree structure, searching for a node in the DOM and modifying nodes are much more efficient compared to the native, raw document files. Making a DOM tree of an XML also makes the data parsing of the XML much more efficient, since the DOM tree explains the structure and the relations of the XML, compared to manually parsing each words.

V. Conclusion

In conclusion, a tree is an important tool in web design, especially in representing Document Object Model. Tree has been a major help in representing DOM from representing a HTML file, to representing the relations of data in XML file.

VI. Acknowledgment

I would like to thank you for Mrs. Harlili, our teacher of IF2120 Matematika Diskrit, 2nd Class for sharing to us her knowledge and wisdoms, my family and my friend for giving me the emotional support in making this paper.

References

[1] Extensible Markup Language (XML), "https://www.w3.org/XML/", World Wide Web Consortium (W3C), retrieved on December 9, 2018.

[2] R. Munir, "Matematika Diskrit Revisi Keempat", Bandung, Indonesia: Informatika Bandung, 2010, pp. 444.

[3] R. Munir, "Matematika Diskrit Revisi Keempat", Bandung, Indonesia: Informatika Bandung, 2010, pp. 457.

[4] Peter-Paul Koch, "*The Document Object Model: an Introduction*", "http://www.digital-web.com/articles/the_document_object_model/", Internet article, retrieved on December 9, 2018.

[5] loc. cit.

[6] "https://www.w3schools.com/xml/books.xml", Sample XML File, w3schools.com, retrieved on December 9, 2018.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 9 Desember 2017

[illegible]

Kevin Fernaldy 13516109