The structure and mechanics of the DOM

Hanako Sato

To add movement to a web page using JavaScript, you need to manipulate the DOM.
However, the DOM is often difficult to understand intuitively, and even when you read an explanation, it is often not very clear. This is the case with me.

There are many ways to manipulate the DOM, but first it is important to understand how the DOM works and how it is structured.

DOM

DOM stands for 'Document Object Model' and represents the content of an entire web page as objects.
The DOM provides an API for handling objects from programming languages such as JavaScript.

Normally, the content of an empty HTML file cannot be manipulated from the JavaScript side.
However, the content of each HTML file can be considered an object, and by accessing specific objects from JavaScript, the appearance of the web page can be changed.

In other words, the DOM is a mechanism for doing the "where" and "what" in HTML from JavaScript.

For example, document.body is an object representing the body tag.
Here is code to change the background color of the HTML page to blue.

```
document.body.style.background = ‘blue’;
```

The structure of the DOM

The DOM has the following features

- A hierarchical structure known as the DOM tree
- Each element of HTML is represented by a node

The DOM tree

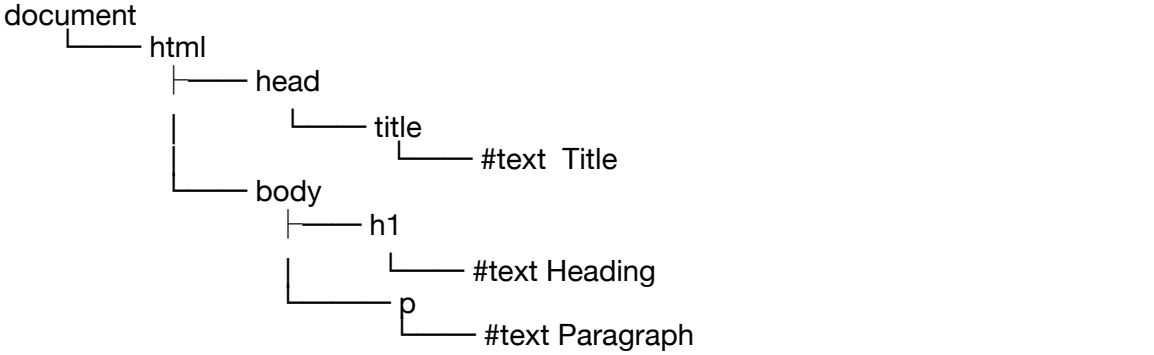
The DOM has a hierarchical structure, like the properties of a branching tree.
The structure is called the DOM tree.

Given the following HTML document.

```
<html>
<head>
  <title>Title</title>
</head>
<body>
  <h1>Heading</h1>
  <p>Paragraph</p>
```

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

The DOM thus represents HTML tags as a DOM tree.



All HTML tags such as <html>, <body> and <h1> are objects.

A tree structure is created with document as the top level, html below it, and head and body below that.
The HTML code you normally write is unconsciously created according to such a tree structure.

Node

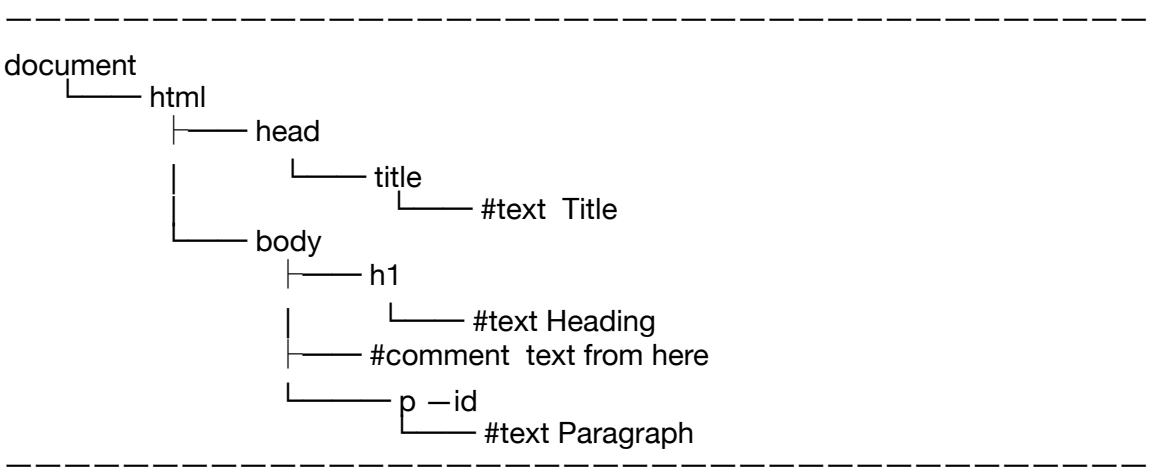
Each element in HTML is called a node in the DOM tree.
Nodes can be of the following types.

Node type	Description
Document node	Document object representing the whole
Element node	Object representing the element
Attribute node	Object representing the text in the element
Comment node	Objects representing comments

See the following HTML document.

```
<html>
<head>
  <title>Title</title>
</head>
<body>
  <h1>Heading</h1>
<! -- text from here -->
  <p id=“text”>Paragraph</p>
</body>
</html>
```

The above code is represented in the DOM tree as.



The document at the top level is the document node.
Element nodes represent the elements of the HTML document, such as head, body, h1 and p.

In addition, text within an element generates a text node, labelled #text.
The id associated with an element is an attribute node.
If there is a comment, a comment node is created, labelled #comment.

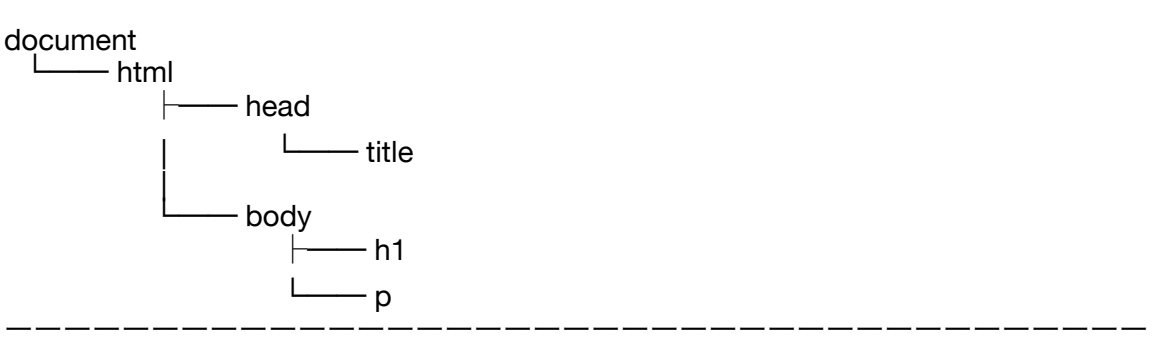
In this way, all elements in HTML are represented as nodes.

Node relationships

A relationship exists between each node depending on its position in the DOM tree.
There are the following types of nodes.

- Root node
- Parent node
- Child node
- Descendant node
- Sibling nodes

See the following DOM tree.



The top-level node that encompasses all nodes is called the root node. Here, document is the root node.

html is a child node of document and a parent node of head and body.
Also, h1 and p are sibling nodes of each other and are descendant nodes of html.

Blank Nodes

Spaces and line breaks in an HTML document are called blank nodes in the DOM.
Blank nodes are treated as text nodes.

The following two HTML documents are displayed exactly the same way in a browser.

```
————————————————————————————————————————————————————
<div><h2>Heading</h2><p>Paragraph</p></div>
————————————————————————————————————————————————————
<div>
  <h2>Heading</h2>
  <p>Paragraph</p>
</div>
————————————————————————————————————————————————————
```

Normally, when an HTML document is displayed in a browser, blank and line breaks before and after an element are not displayed.
This is because blank is ignored according to certain rules.

However, the DOM tree has a different structure.

```
————————————————————————————————————————————————————
div
  ├── h2
  │     └── #text Heading
  └── p
        └── #text Paragraph
————————————————————————————————————————————————————
div
  ├── blank
  ├── h2
  │     └── #text Heading
  ├── blank
  ├── p
  │     └── #text Paragraph
  └── blank
————————————————————————————————————————————————————
```

It is necessary to understand about blank nodes because the process of retrieving a node in a DOM operation may differ depending on whether there are blanks or not.

conclusion

* DOM captures the content of an entire web page as an object
* Represent an HTML document as a DOM tree with a branching structure
* All elements in HTML are represented as nodes
* Text, blank, and comments are also treated as nodes