DOMとはJavaScriptでhtmlの要素を操作するための仕組みのことだ。

JavaScriptを扱っていく上で、絶対に知らないといけない仕組みのひとつだろう。

このページではDOMの仕組みと使い方について初心者の方でもわかるように解説した。

目次 [hide]

DOMとは何か？

DOMとは「Document Object Model」の略だ。直訳すると、「ドキュメントを物として扱うモデル」になる。プログラムからHTMLやXMLを自由に操作するための仕組みだ。

例えばブラウザに表示される文字の色を変更したり、大きくしたりと、Webページの見た目をプログラムで処理をしたい場合があるだろう、しかし何もしていない状態のHTMLファイルではJavaScriptから手を出す事が出来ない。そこでファイルの特定の部分に目印を付けて「この部分」に「こういう事をしたい」という処理を可能にするための取り決めがDOMである。

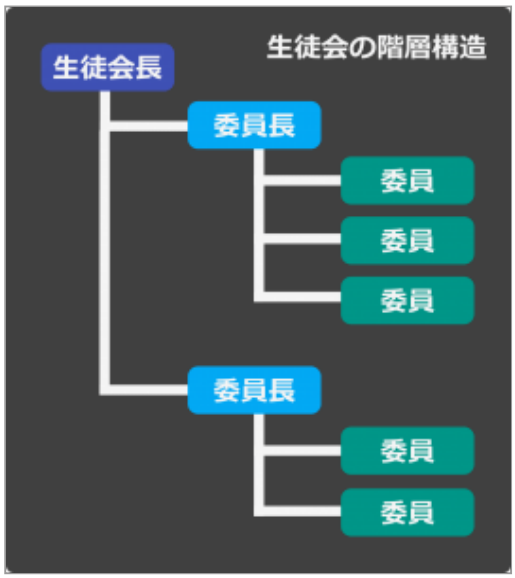
DOMは以下のような特徴をもっている。

- ツリー構造とも呼ばれる階層構造を取る
- それぞれノードという言葉で説明される
- WEBページとJavaScriptなどのプログラミング言語とを繋ぐ

それでは、DOMの特徴について1つ1つ詳細に見ていこう。

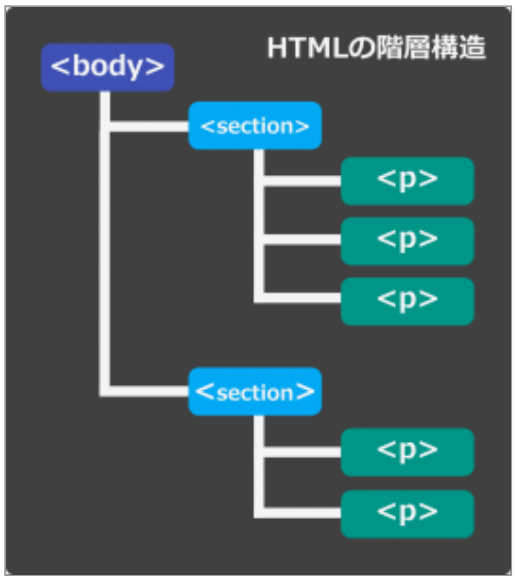
DOMは「階層構造」を取る

階層構造とは組織図のようなものだ。今回は高校の生徒会の組織図を例に出してみる。



生徒会の組織図は、上の図にある通り生徒会長を頂点として、下に何人かの委員長と、その更に下に何人かの委員が所属して階層構造が作られている。

次に、HTMLの階層構造を見てみよう。



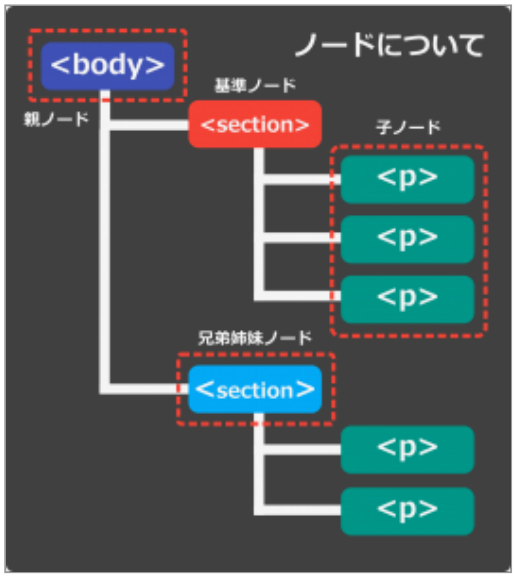
<body>を頂点として、下にいくつかの<section>と、そのさらに下にいくつかの<p>で構成されている。

これはHTMLで階層構造を構築した場合の一例だ。この階層構造を定義しているものがDOMと呼ばれる仕組みを使っていることになる。

各要素は「ノード」という単語を用いて表現される

DOMで必ず出てくる用語「ノード」について補足しておく。

- ノード
- 子ノード
- 親ノード
- 兄弟姉妹ノード



上の図にあるように、ノードとは各要素(HTMLではエレメントやタグという)自体のことを表す。

特定のノードを基準としたときに、その上にあるノードを「親:parent」ノードと表現し、その下にあるノードを「子:childまたはchildren」ノードと表現する。

また同一階層にあるノードのことを「兄弟姉妹:siblings」ノードと表現する。

例えば「そのタグの子ノード全体を取得して、その親ノードから削除する」のような使い方をする。

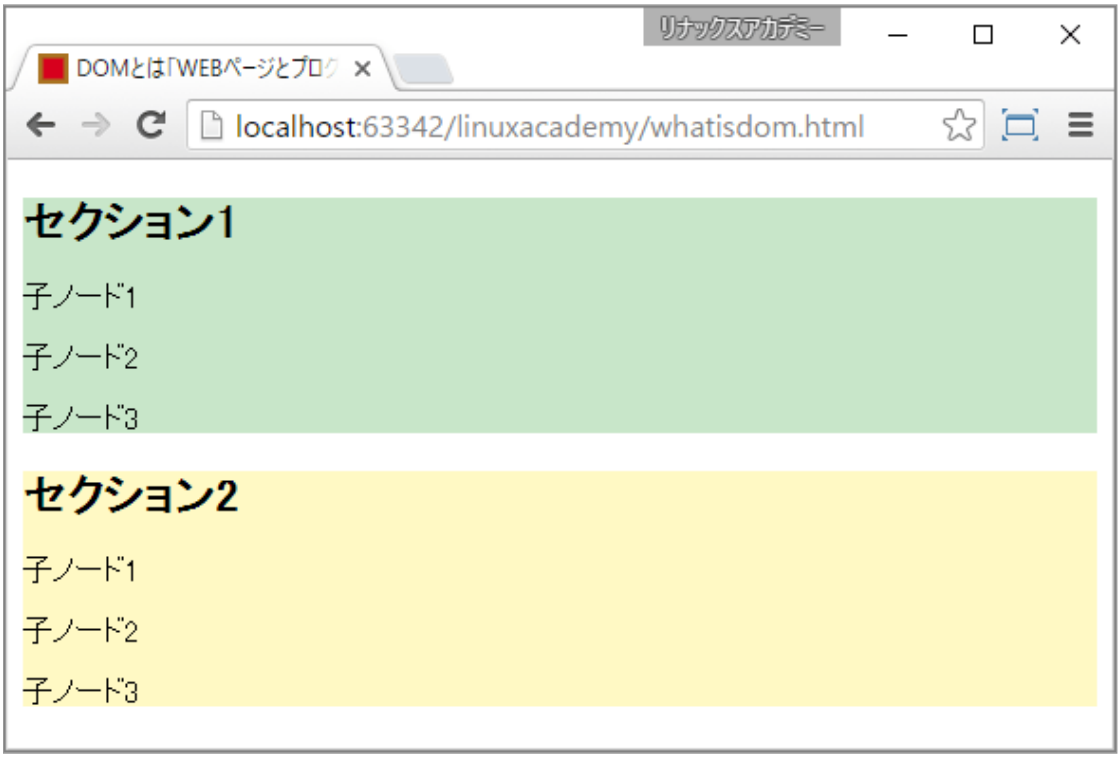
DOMは「WEBページとプログラミング言語を繋ぐ役割を持つ」

それでは、実際のソースコードを見てみよう。DOMに沿った記述を行ったHTMLファイルをJavaScriptから操作してみる。

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <link rel="shortcut icon" type="image/x-icon" href="favicon.ico">
    <title>DOMとは「WEBページとプログラミング言語を繋ぐ役割を持つ」</title>
    <style>
        #section-1 {
            background-color: #C8E6C9;
        }
        #section-2 {
            background-color: #FFF9C4;
        }
    </style>
</head>
<body>
<section id="section-1">
    <h2>セクション1</h2>
    <p>子ノード1</p>
    <p>子ノード2</p>
    <p>子ノード3</p>
</section>
<section id="section-2">
    <h2>セクション2</h2>
    <p>子ノード1</p>
    <p>子ノード2</p>
    <p>子ノード3</p>
</section>
</body>
<script>
    // ここにJavaScriptコードを書いていく
    // ...
</script>
</html>
```

今回使用するソースコードは、簡略化のためにHTMLとCSSとJavaScriptが全て1つのページに書いてあるものにしてある。

これをWEBブラウザで表示すると図4のような表示になる。



ID名からノードを取得して、操作する

ここでは詳しく解説しないが、ID名とはタグにつける一意の名前のことだ。CSSを勉強したときや、JavaScriptの基本で既に出てきているだろう。

今回のサンプルでは「section-1」と「section-2」がID名に該当する。

ID名を指定するメソッドは以下のように定義されている。

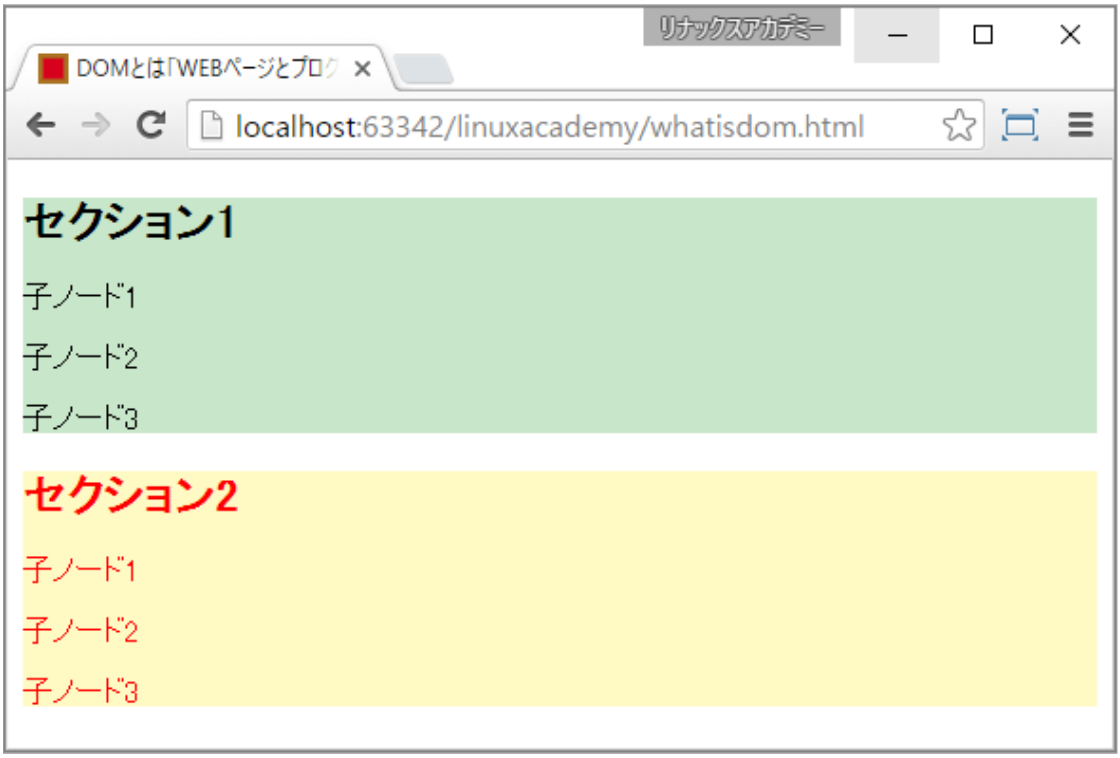
```
document.getElementById(id);
```

idはタグについているID名だ。

これを用いて「section-2」を取得して、その文字色を赤色に変更してみよう。

```
1 document.getElementById('section-2').style.color = 'red';
```

このJavaScriptをブラウザで実行すると図5のようになる。



黄色の背景に入っている(section-2およびその子ノード全体の)文字色が赤くなっているのが確認できる。

子ノードを取得して、操作する

子ノードを指定して、操作をしてみよう。

子ノードを指定するメソッドは以下のように定義されている。

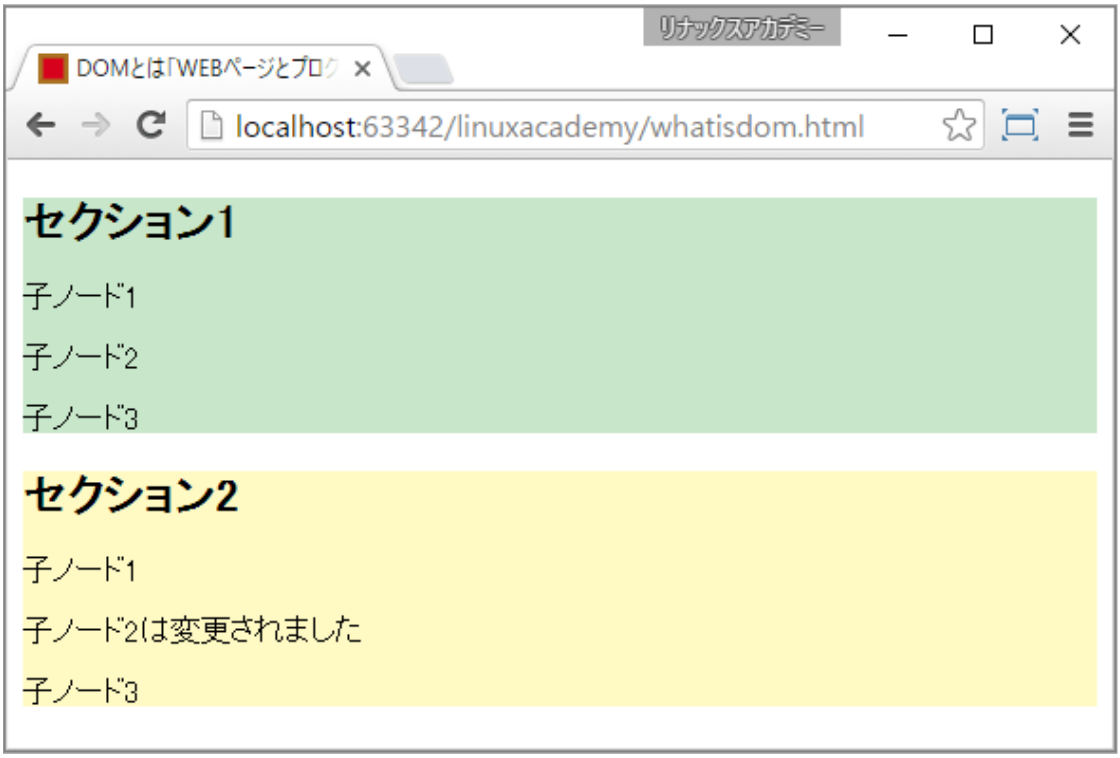
```
var nodeList = elementNodeReference.childNodes;
```

- elementNodeReferenceは基準ノードを指している
- nodeList は基準ノードについている子ノード全てを指す

これを用いて「section-2」の子ノードのうち2番目の子ノードの文字列を変更してみよう。

```
1 var baseElement = document.getElementById('section-2');
2 var section2node2 = baseElement.childNodes[5];
3 section2node2.innerHTML = '子ノード2は変更されました';
```

このJavaScriptをブラウザで実行すると図6のようになる。



セクション2の子ノード2の文字列が変更されているのが確認できる。

補足：baseElement.childNodes[5];は2番目のノードを取得したいのになぜ「5」なのか？

Chromeブラウザのchildnodesの実装が、ノードとノードの間に「空白のノード」を差し込むものになっているため、一見奇妙な配列の添え字になっている。

例えば、Chromeブラウザで1番目のノードを取得したい場合、添え字は「3」になる。

親ノードを取得して、操作する

親ノードを指定して、操作をしてみよう。

親ノードを指定するメソッドは以下のように定義されている。

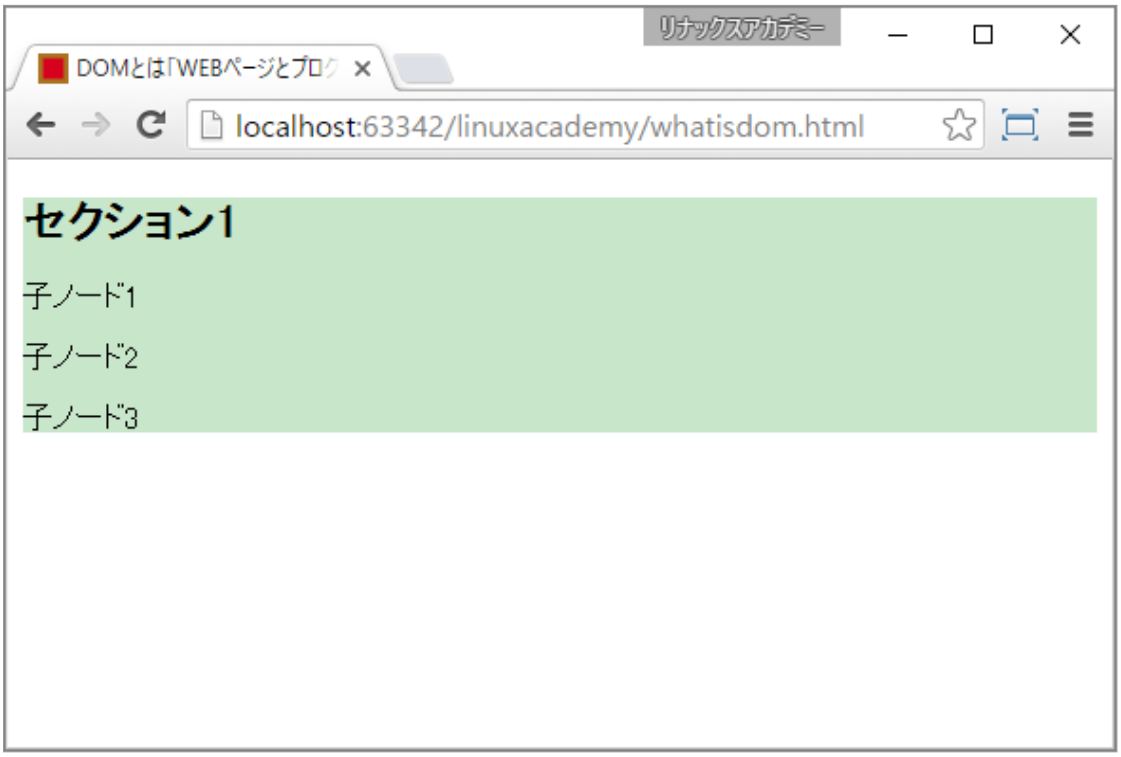
```
var parentNode = elementNodeReference.parentNode
```

- elementNodeReference ... 基準ノード
- parentNode ... 基準ノードの親ノード

これを用いて「section-2」の親ノードを取得して、その親ノードの子ノード(つまりsection-2)全てを削除してみよう。

```
1 var baseElement = document.getElementById('section-2');
2 var parentNode = baseElement.parentNode;
3 parentNode.removeChild(baseElement);
```

このJavaScriptをブラウザで実行すると図7のようになる。



セクション2が全て消えているのが確認できる。

このように、DOMに沿った書き方で定義された「ノード」を通じてJavaScriptからHTMLを操作することができることが理解できたと思う。

これが「DOM」が「WEBページを表示する言語であるところのHTML」と「プログラミング言語であるところのJavaScript」を繋ぐ役目を持つということだ。

まとめ

JavaScriptで登場するDOMについて簡単にご紹介してきたが、いかがだっただろうか？

階層構造であること、指定したノードやその親や子などを自由にJavaScriptから操作ができることを理解しておこう。