Document Object Model

views the web page as a collection of HTML elements and text nodes arranged as a tree

```
<html>
<p>Hello</p>
<p>Welcome to <b>CE212</b>!</p>
</html>
```

In this example

The root is an `html` element
It has two children, both `p` elements

The first `p` element has one child, a text node with content `Hello`

The second `p` element has three children, a text node with content `Welcome to ` , a `b` element, and a text node with content `!`

The `b` element has one child, a text node with content `CE212`

To add or manipulate content we need to locate an existing element in the document; this is normally done using an `id` attribute

```
<div id="x"> </div>
```

```
var div =
document.getElementById("x");
```
selects the element (assuming the above HTML fragment exists somewhere in the document)

[When running Javascript in a browser the variable `document` can be used to refer to the HTML document]

```
var p = document.createElement("p");
```
creates a new `p` element:
```
<p></p>
```

We need to add it to the document as a child of an existing element
```
div.appendChild(p);
```

```
<div id="x">
<p></p>
</div>
```

We can create a text node and add it as a child to the `p` element
```
var tn =
document.createTextNode("Welcome to
");
p.appendChild(tn);
```

```
<div id="x">
```

```
<p>Welcome to </p>
</div>
```

It is not necessary to use a variable if we do not need to refer to the element/node again; we could have used

```
p.appendChild(
  document.createTextNode("Welcome to
"));
```

We can add two more children to the `p` element:

```
var b = document.createElement("b");
p.appendChild(b);
p.appendChild(document.createTextNode
("!"));
```

```
<div id="x">
<p>Welcome to <b></b>!</p>
</div>
```

[`appendChild` adds the child after any existing children; there are ways to insert it in front of exsting children]

Finally we can add text content to the `b` element

```
b.appendChild(document.createTextNode
("CE212"));
```

```
<div id="x">
<p>Welcome to <b>CE212</b>!</p>
</div>
```

We can maniuplate the HTML attributes of an element:

```
p.setAttribute("class", "highlight");
```

```
<div id="x">
<p class = "highlight">Welcome to
<b>CE212</b>!</p>
</div>
```

Any CCS rule for `#highlight` would now be applied to the element