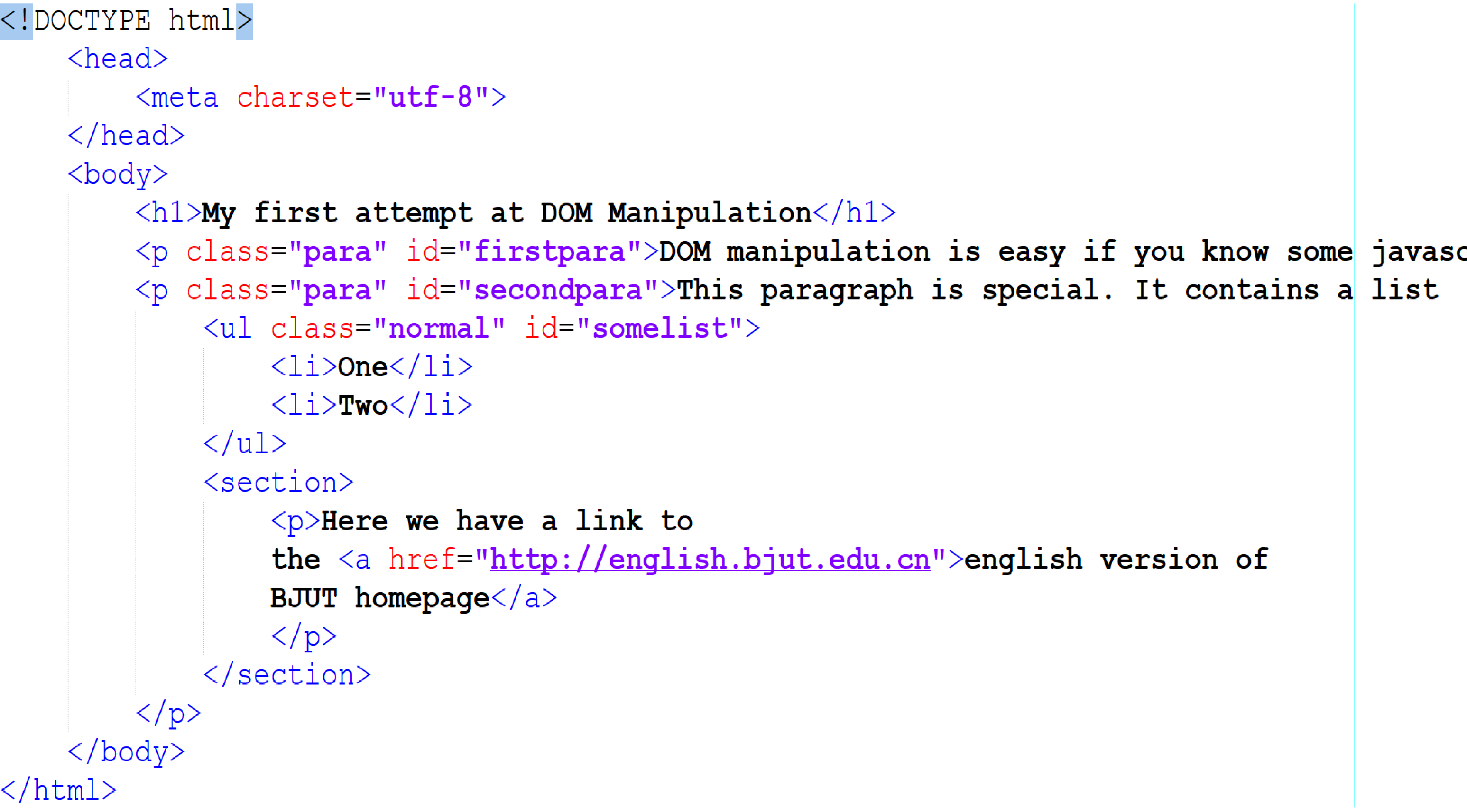
DOM & JAVASCRIPT

IMPORTANT OBJECTS

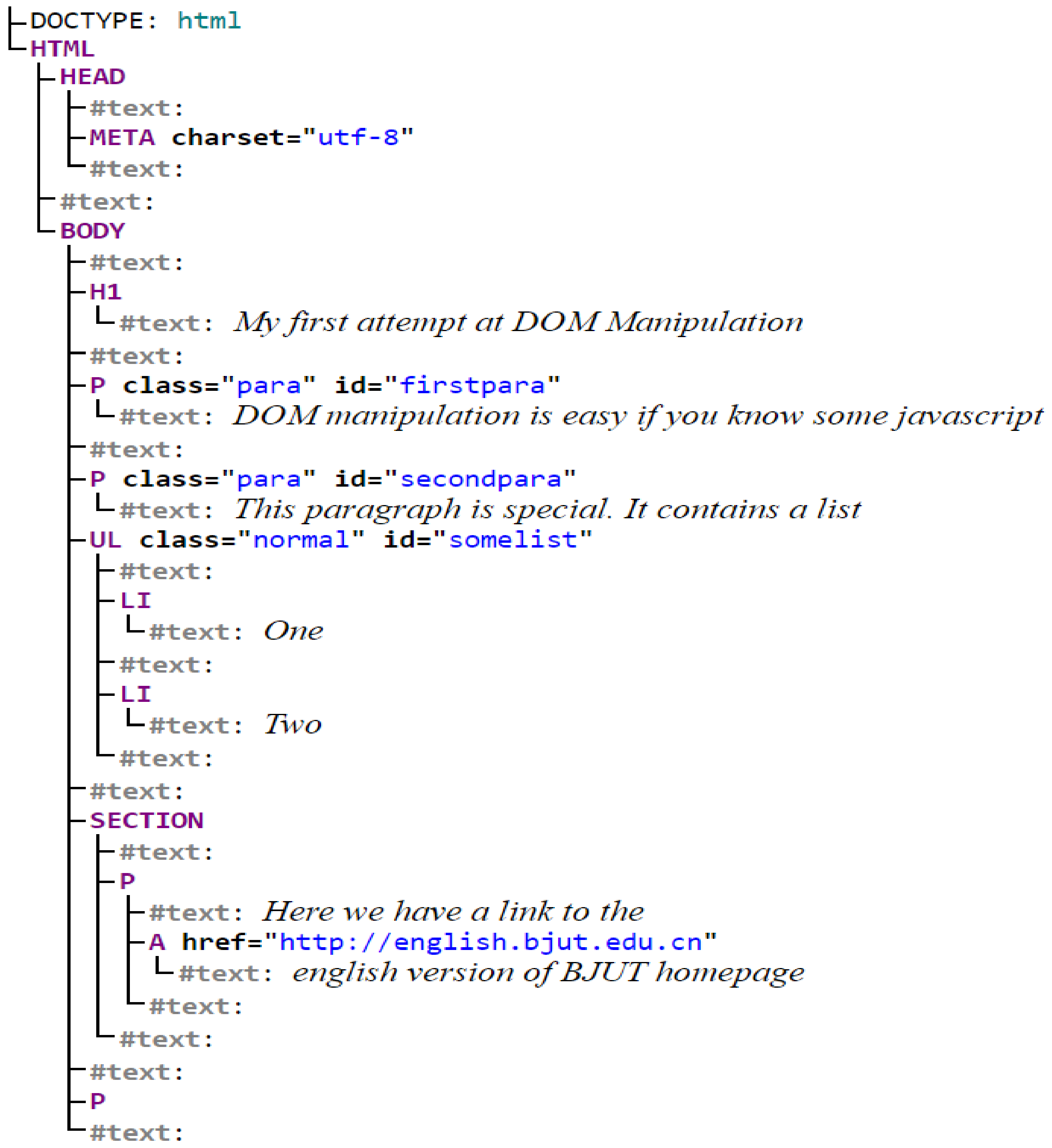
- Window Object – The browser tab that the web page is loaded into. Contains properties such as `Window.innerWidth` and `Window.innerHeight`
- Navigator Object – Represents the state and identity of the browser (i.e., user-agent). Retrieves things like user's preferred language, media stream from the webcam, etc.
- Document Object – Represents the actual page loaded into the window. Used to manipulate HTML and CSS that comprises the document.



```html
<!DOCTYPE html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>My first attempt at DOM Manipulation</h1>
        <p class="para" id="firstpara">DOM manipulation is easy if you know some javasc
        <p class="para" id="secondpara">This paragraph is special. It contains a list
            <ul class="normal" id="somelist">
                <li>One</li>
                <li>Two</li>
            </ul>
            <section>
                <p>Here we have a link to
                the <a href="http://english.bjut.edu.cn">english version of
                BJUT homepage</a>
                </p>
            </section>
        </p>
    </body>
</html>
```

DOCUMENT OBJECT MODEL

- Element Node
- Root Node
- Child Node
- Descendant Node
- Parent Node
- Sibling Nodes
- Text Node



```
├DOCTYPE: html
└HTML
  ├HEAD
  │ ├#text:
  │ ├META charset="utf-8"
  │ └#text:
  ├#text:
  └BODY
    ├#text:
    ├H1
    │ └#text: My first attempt at DOM Manipulation
    ├#text:
    ├P class="para" id="firstpara"
    │ └#text: DOM manipulation is easy if you know some javascript
    ├#text:
    ├P class="para" id="secondpara"
    │ └#text: This paragraph is special. It contains a list
    ├UL class="normal" id="somelist"
    │ ├#text:
    │ ├LI
    │ │ └#text: One
    │ ├#text:
    │ ├LI
    │ │ └#text: Two
    │ └#text:
    ├#text:
    ├SECTION
    │ ├#text:
    │ ├P
    │ │ ├#text: Here we have a link to the
    │ │ ├A href="http://english.bjut.edu.cn"
    │ │ │ └#text: english version of BJUT homepage
    │ │ └#text:
    │ └#text:
    ├#text:
    ├P
    └#text:
```

BASIC MANIPULATION

- To manipulate an element, the steps are:
  - Select the element

```
document.querySelector()
```

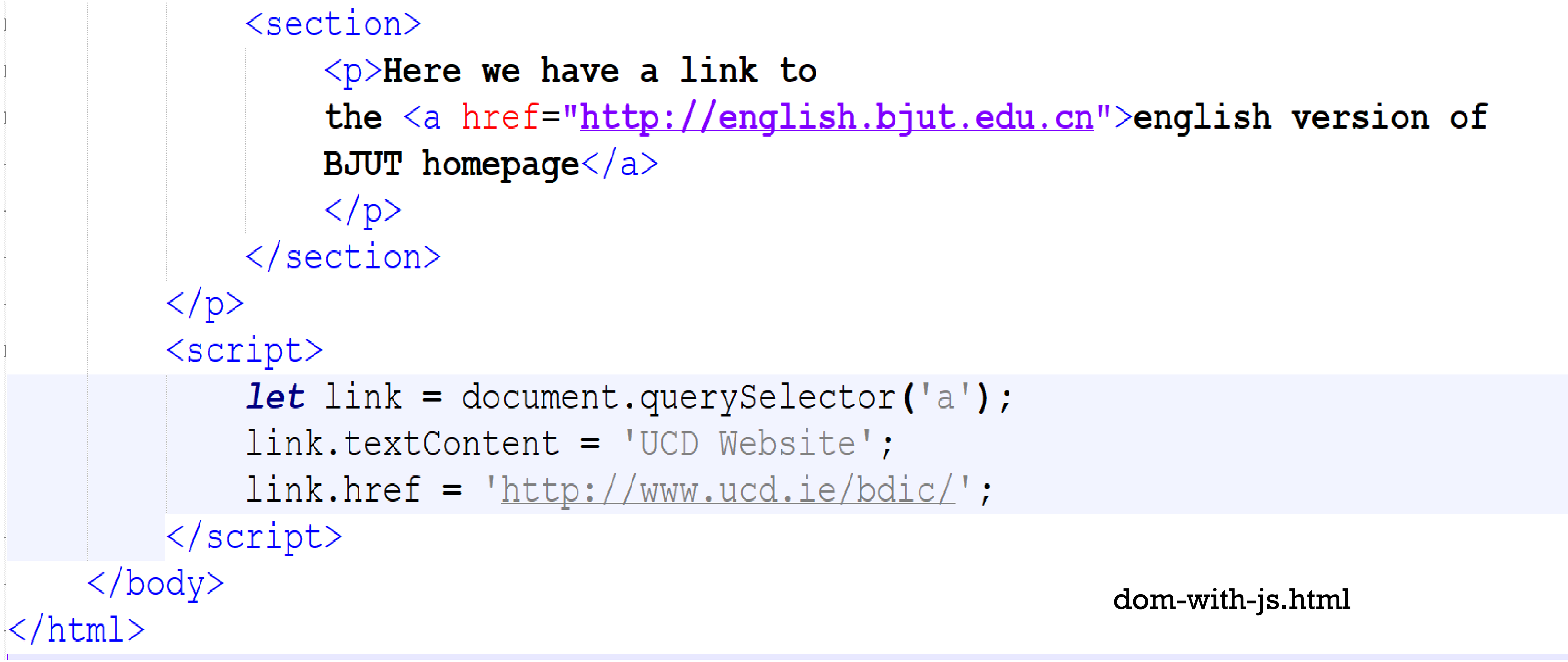
  - Store the reference inside a variable

```
let myVar = document.querySelector(‘a’);
```

  - Change whatever property you want

```
myVar.href = ‘https://some.website.url/’;
```

SAMPLE MANIPULATION OF LINK



```
            <section>
                <p>Here we have a link to
                the <a href="http://english.bjut.edu.cn">english version of
                BJUT homepage</a>
                </p>
            </section>
        </p>
        <script>
            let link = document.querySelector('a');
            link.textContent = 'UCD Website';
            link.href = 'http://www.ucd.ie/bdic/';
        </script>
    </body>
</html>
```

dom-with-js.html

OLDER METHODS

- **document.getElementById()**

e.g., `let myVar = document.getElementById(“firstpara”);`

- **document.getElementsByTagName()**
- **e.g.,** `let myVarArray = document.getElementsByTagName(“p”);`

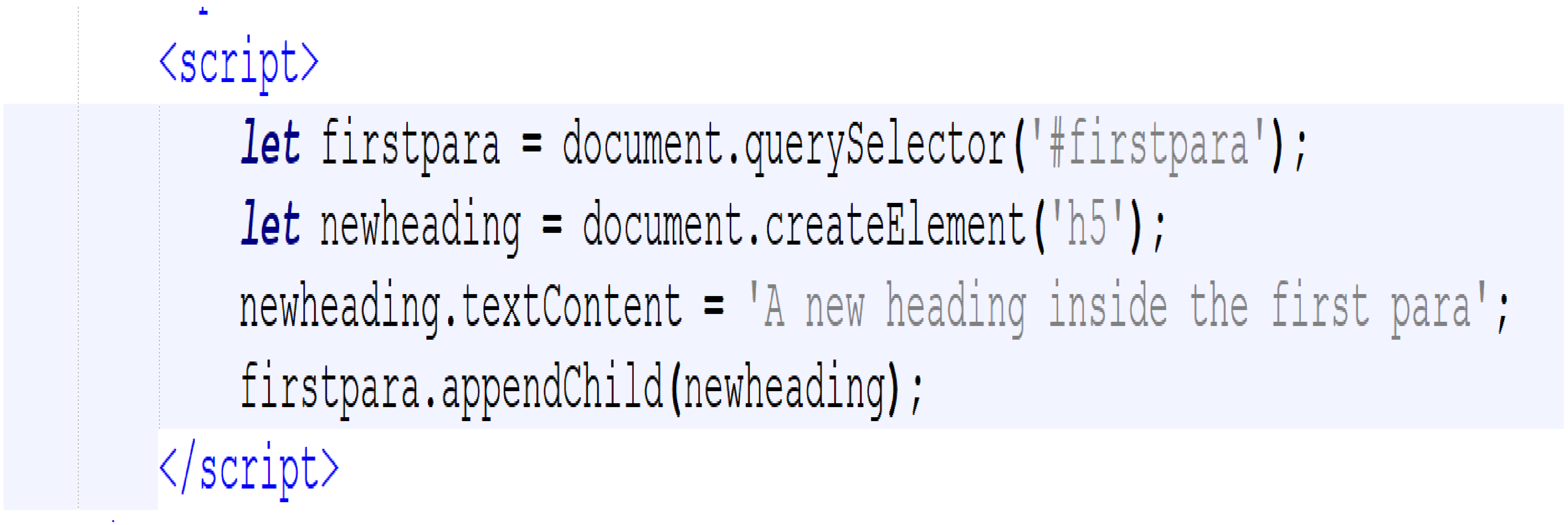
- **Equivalent newer method: document.querySelectorAll()**

e.g., `let myVarList = document.querySelectorAll(“p”);`

CREATING AND PLACING NEW NODES

- Steps involved in creating new nodes
  - Select the parent node, using `document.querySelector()`
  - Create a new element using `document.createElement()`
  - Give it some content (say, `textContent or href or` any other property)
  - Append new element to the parent node using `appendChild()`

SAMPLE CREATION

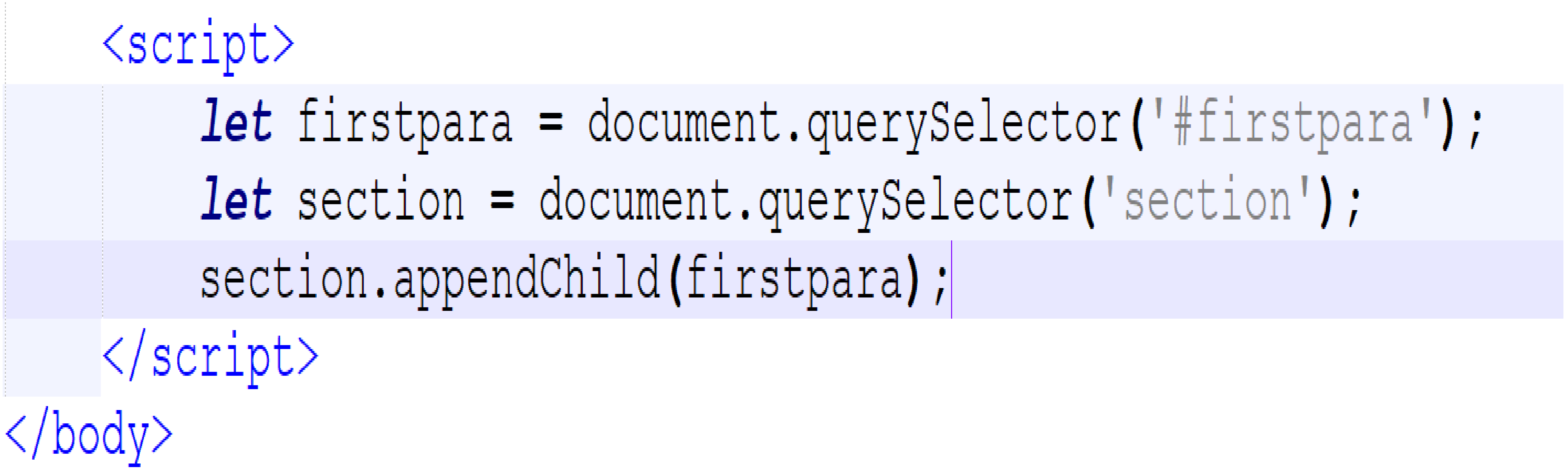

```
<script>
    let firstpara = document.querySelector('#firstpara');
    let newheading = document.createElement('h5');
    newheading.textContent = 'A new heading inside the first para';
    firstpara.appendChild(newheading);
</script>
```

dom-new-node.html

MANAGING AND REMOVING NODES

- Sometimes we want to move nodes around the page or delete them entirely
- The steps involved in moving nodes are:
  - Get a reference to the node you want to move using `document.querySelector()`
  - Get a reference to the new parent node using `document.querySelector()`
  - Use `appendChild` to move the node to the parent
  - NOTE: This will move the actual node. If you want to make a copy, use `cloneNode()` on the node you want to copy

SAMPLE MOVEMENT



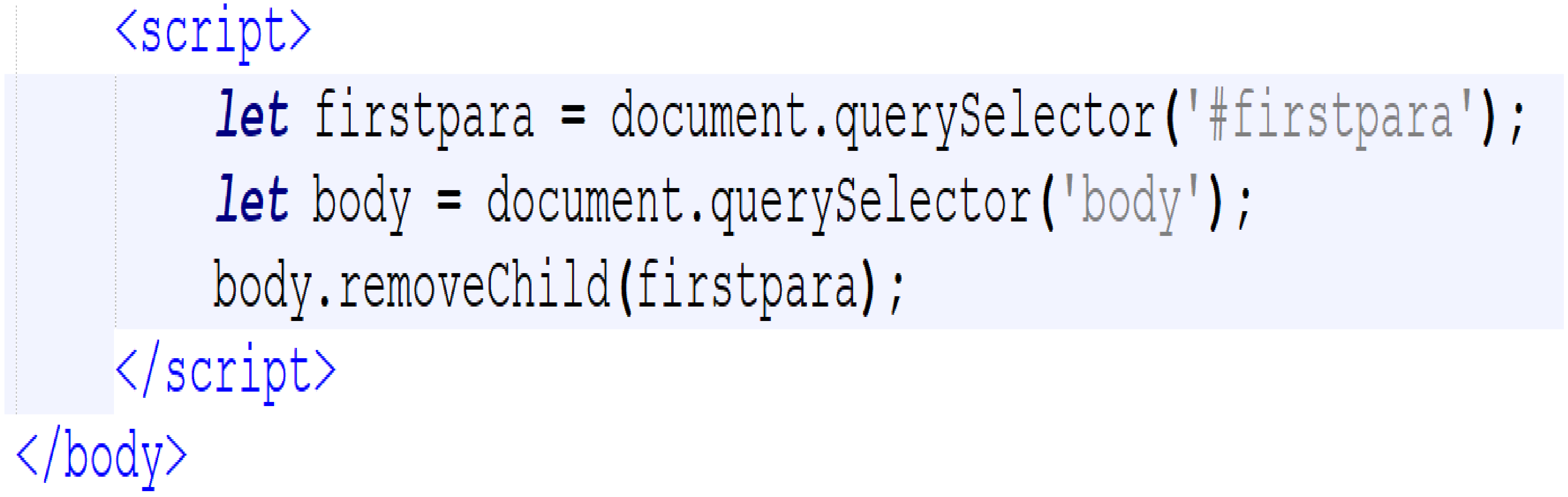
```
    <script>
        let firstpara = document.querySelector('#firstpara');
        let section = document.querySelector('section');
        section.appendChild(firstpara);
    </script>
</body>
```

dom-move-node.html

DELETING NODES

- Steps to delete nodes are very similar
- Get a reference to the node you want to remove using `document.querySelector()`
- Get a reference to the parent node using `document.querySelector()`
- Call `removeChild` on the parent, to remove the child node

SAMPLE REMOVAL



```
    <script>
        let firstpara = document.querySelector('#firstpara');
        let body = document.querySelector('body');
        body.removeChild(firstpara);
    </script>
</body>
```

dom-remove-node.html

OTHER WAYS TO DELETE

- In modern browsers, a node can also delete iself

```
let firstpara = document.querySelector('#firstpara’);
firstpara.remove();
```

- In older browsers, you must have a reference to the parent

```
firstpara.parentNode.removeChild(firstpara);
```

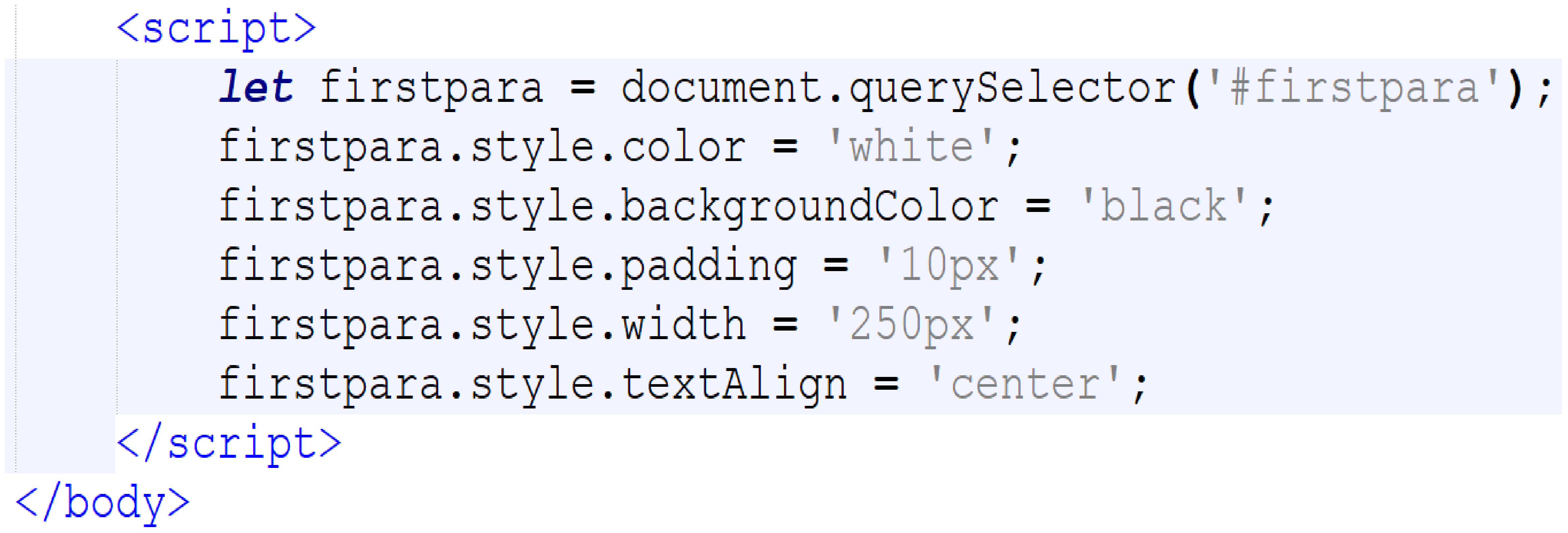
ADD STYLE TO NODES

- Steps to add some styling information

- Get a reference to the node you want using `document.querySelector()`
- Add style information to the `style` property of the node

- NOTE: In Javascript, the style names are camelCase instead of the CSS names which are hyphenated
- `backgroundColor` instead of `background-color`

SAMPLE STYLE

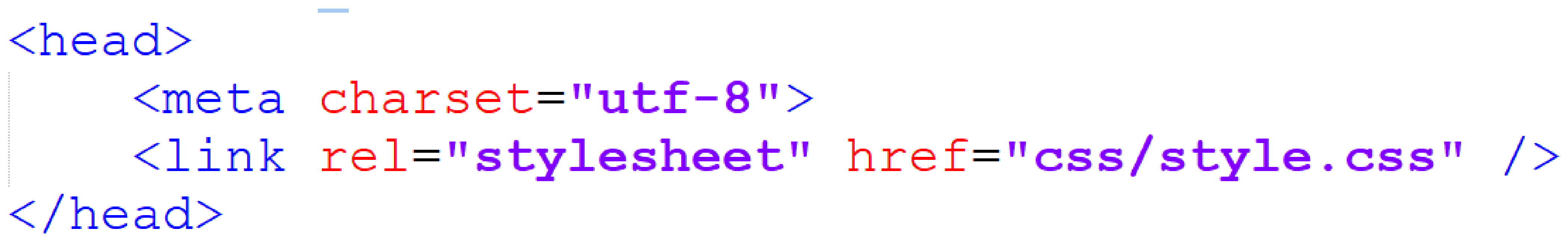


```
    <script>
        let firstpara = document.querySelector('#firstpara');
        firstpara.style.color = 'white';
        firstpara.style.backgroundColor = 'black';
        firstpara.style.padding = '10px';
        firstpara.style.width = '250px';
        firstpara.style.textAlign = 'center';
    </script>
</body>
```

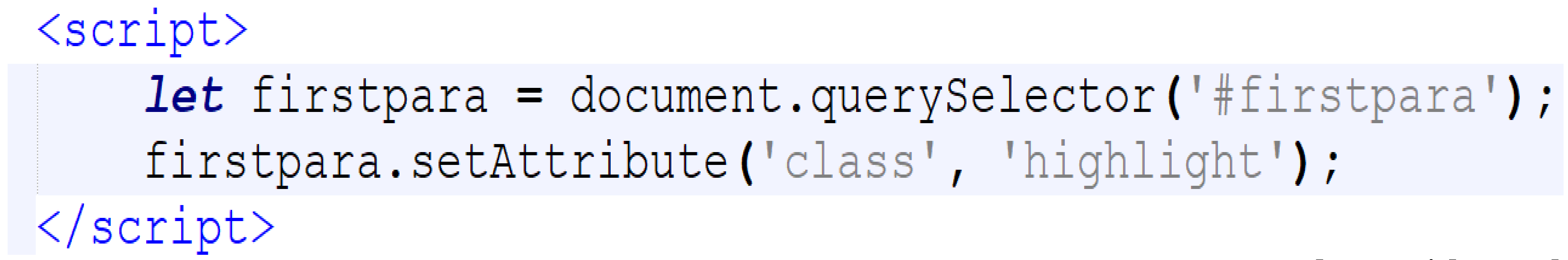
dom-add-style.html

CLEANER WAY

- Create a stylesheet (the way you did in the CSS class)
- Get a reference to the node you want to style
- Apply the styling rule you want



```html
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="css/style.css" />
</head>
```



```html
<script>
    let firstpara = document.querySelector('#firstpara');
    firstpara.setAttribute('class', 'highlight');
</script>
```

dom-with-css.html

USING THE WINDOW OBJECT

- Get a reference to the node you want to change
- Get a reference to the window object
- Change the node, according to the properties from the Window object

SAMPLE – CHANGE DIV AS PER WINDOW SIZE

```
<script>
    let div = document.querySelector('div');
    let winWidth = window.innerWidth;
    let winHeight = window.innerHeight;

    div.style.width = winWidth + 'px';
    div.style.height = winHeight + 'px';
</script>
```

dom-window.html   dom-window-script.html

ANOTHER WAY

```
<script>
    window.onresize = function(){
        let div = document.querySelector('div');
        let winWidth = window.innerWidth;
        let winHeight = window.innerHeight;

        div.style.width = winWidth + 'px';
        div.style.height = winHeight + 'px';
    }
</script>
```

dom-window-resize.html

USEFUL EVENTS

- onresize (Window)
- onclick
- onblur
- onchange
- onclose (Window)
- ondblclick
- onerror
- onfocus
- onload

EXERCISE (TO DO IN CLASS)

- Download the shopping-list.html file
- Create three variables to hold references to <ul>, <input> and <button>
- Create a function that runs when the button is clicked
- Inside the function body, do the following:
  - Store the current value of input element in a variable
  - Empty the input value by setting its value to an empty string "   "
  - Create new elements: a list item <li>, <span>, <button> and store references in variables
  - Append the span and button as children of the list item
  - Set the text content of the span to the input element value you saved earlier, and the text content of the button to 'Delete'
  - Append the list item as a child of the list.
  - Attach an event handler to the delete button, so that when clicked it will delete the entire list item it is inside