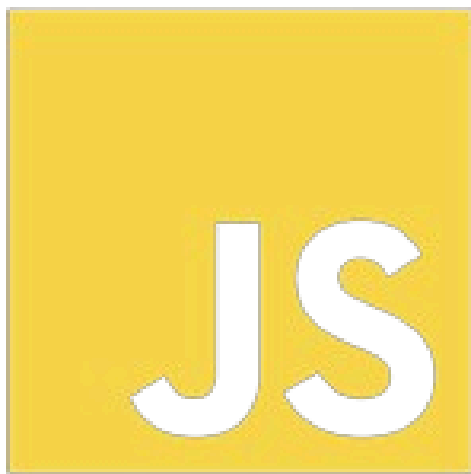


JavaScript DOM Manipulation



Hasan Raza
Junior Web Developer



Window Object

*When we open a browser tab, we always have the **window object** by default. It represents the browser's open window and serves as a global object with many **properties and methods**.*



Window

The diagram illustrates the browser object model hierarchy using a series of nested rectangles. The outermost rectangle is orange and labeled 'Window'. Inside it is a green rectangle labeled 'Document'. Inside 'Document' is a blue rectangle labeled 'HTML'. Inside 'HTML' are two dark blue rectangles: 'Head' on the left and 'Body' on the right. The rectangles are nested to show that 'Window' contains 'Document', which contains 'HTML', which in turn contains 'Head' and 'Body'.

Document

HTML

Head

Body

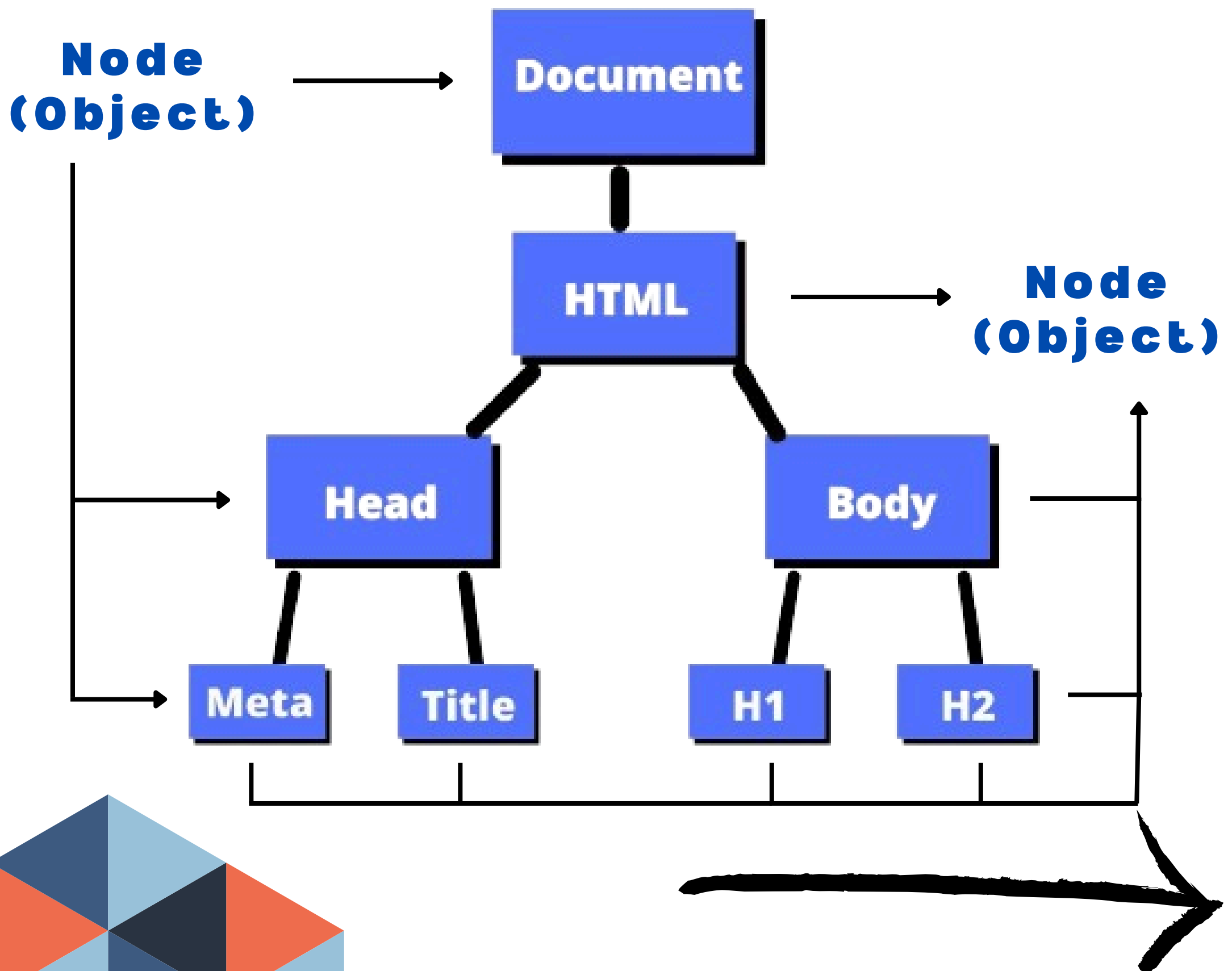


What is DOM?

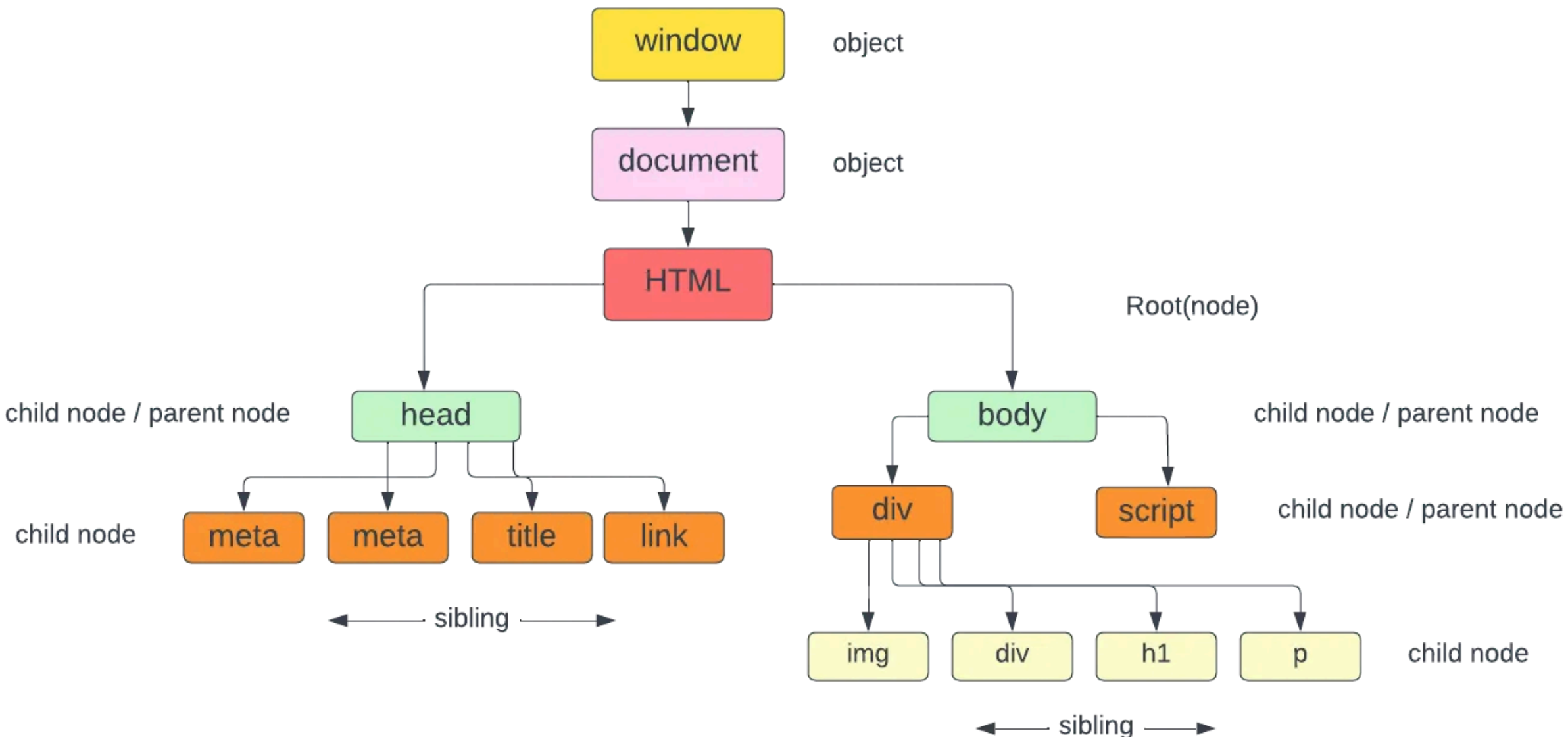
In JavaScript, we can access our entire HTML code. Automatically, our HTML elements are converted into JavaScript objects. This special object is called the document, and it is available within our window object. This document contains our entire HTML code. So its simple definition is:



“When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.”



DOM Tree



*Each element can have one or more **child elements** and exactly one **parent element**. Multiple children are **siblings** to each other.*



Selecting Elements

To manipulate elements, we first need to select them. There are 5 methods to access element from HTML:

1. **getElementById():**

Selects a single element by its ID.

```
const element = document.getElementById('myElement');
```



2. **getElementsByClassName():**

Selects all elements with a specific class.



```
const elements = document.getElementsByClassName('myClass');
```

3. **getElementsByTagName():**

Selects all elements with a specific tag name.



```
const elements = document.getElementsByTagName('div');
```



4. **querySelector():**

Selects the first element that matches a CSS selector.

```
const element = document.querySelector('.myClass');
```

5. **querySelectorAll():**

Selects all elements that match a CSS selector.

```
const elements = document.querySelectorAll('.myClass');
```



Insert Elements

There are two steps to inserting elements in JavaScript:

- 1. First, we create the element.*
- 2. Then, we add it to the DOM.*

Creating an element:

The `createElement` method is used to create a new HTML element.

```
const newDiv = document.createElement('div');
```



Creating Text Content:

To add text content to an element, you can create a text node using the `createTextNode` method.

```
const textContent = document.createTextNode('Hello, World!');
```

Adding Element:

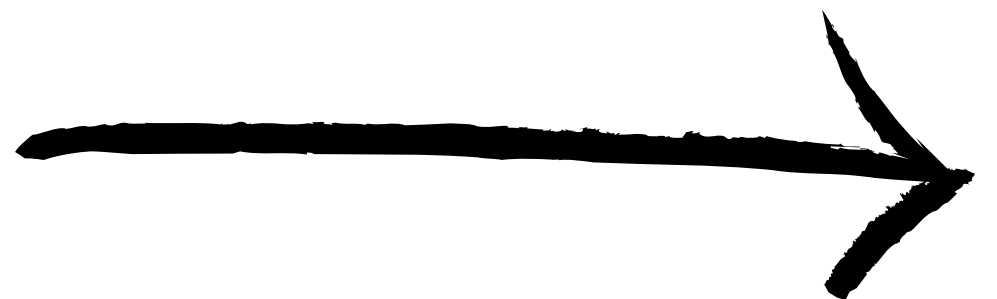
To display element to our screen, there are four main methods:

1. *`node.append(el)`*

2. *`node.prepend(el)`*

3. *`node.before(el)`*

4. *`node.after(el)`*



1. **node.append(el):**

Adds a node as the last child of a specified parent node.

```
const newDiv = document.createElement('div');  
newDiv.innerText = 'Appended Child';  
document.body.append(newDiv);
```

2. **node.prepend(el):**

Adds a node as the first child of a specified parent node.

```
const newDiv = document.createElement('div');  
newDiv.innerText = 'Prepended Child';  
document.body.prepend(newDiv);
```



3. **node.before(el):**

Inserts a node before a specified node(outside).

```
const newDiv = document.createElement('div');
newDiv.innerText = 'Inserted Before';
const referenceNode = document.getElementById('reference');
referenceNode.before(newDiv);
```

4. **node.after(el):**

Inserts a node after a specified node.

```
const newDiv = document.createElement('div');
newDiv.innerText = 'Inserted After';
const referenceNode = document.getElementById('reference');
referenceNode.after(newDiv);
```



Properties

1. innerHTML:

*Sets or gets the **HTML content** of an element.*

```
javascript

document.getElementById('container').innerHTML = '<p>New HTML content</p>';
```

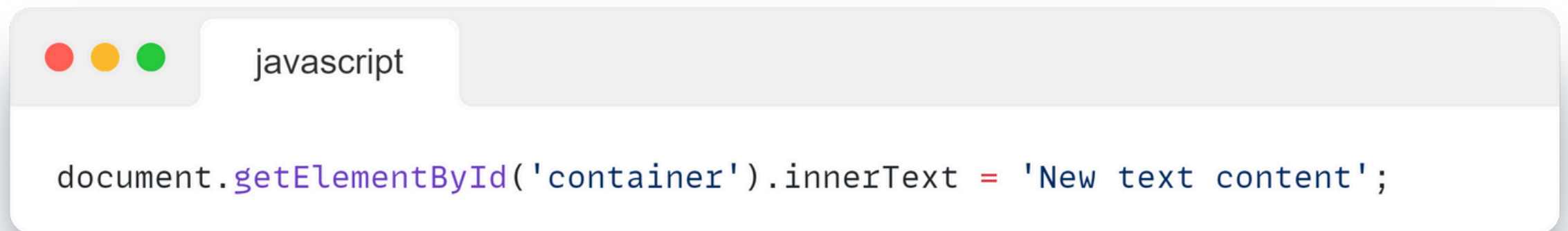
```
html

<div id="container">
  <p>New HTML content</p>
</div>
```



2. innerText:

*Sets or gets the **text content** of an element, excluding hidden elements.*

A code editor window with a light gray header bar containing three colored circles (red, yellow, green) and a tab labeled 'javascript'. The main area is white and contains a single line of JavaScript code.

```
document.getElementById('container').innerText = 'New text content';
```

A code editor window with a light gray header bar containing three colored circles (red, yellow, green) and a tab labeled 'html'. The main area is white and contains a single line of HTML code.

```
<div id="container">New text content</div>
```



3. `textContent`

Sets or gets the `textContent` of an element, including hidden elements.

● ● ●

javascript

```
document.getElementById('container').textContent = 'New text content';
```

● ● ●

html

```
<div id="container">New text content</div>
```



4. tagName:

*Gets the **tag name** of an element.*

```
javascript

const tagName = document.getElementById('container').tagName;
console.log(tagName); // Output: DIV
```

5. className:

*Sets or gets the **class name(s)** of an element.*

```
javascript

document.getElementById('container').className = 'new-class';
```

```
html

<div id="container" class="new-class">Container</div>
```



Modifying Elements

We can *change* the attributes, styles, and content of elements.

1. Changing attributes:

to *set/ change* the attribute value

```
const element = document.getElementById('myElement');  
element.setAttribute('class', 'newClass');
```

↓
attribute

↓
value



2. Getting Attribute:

*to **get** the attribute value from HTML:*

```
const element = document.querySelector("p");  
console.log(element.getAttribute("class"));  
// Output: para(class in HTML)
```

value

attribute

3. Changing Styles:

*to change any **css style**:*

```
element.style.color = 'blue';  
element.style.fontSize = '20px';
```



CSS

JavaScript

color



color

background_color



backgroundColor

font_size



fontSize

*We always use **camel case** in JavaScript.*

4. Remove Elements:

*to **remove** an element:*



```
const element = document.getElementById('container');  
element.remove();
```





Thank you for watching!

Follow me for more...



Hasan Raza



Hasanraza09

