



Day 34

“Web Development + Security”

JavaScript DOM:

What Is the DOM?

The DOM (Document Object Model) is a tree-like structure representing all the elements of an HTML page.

When a browser loads a webpage:

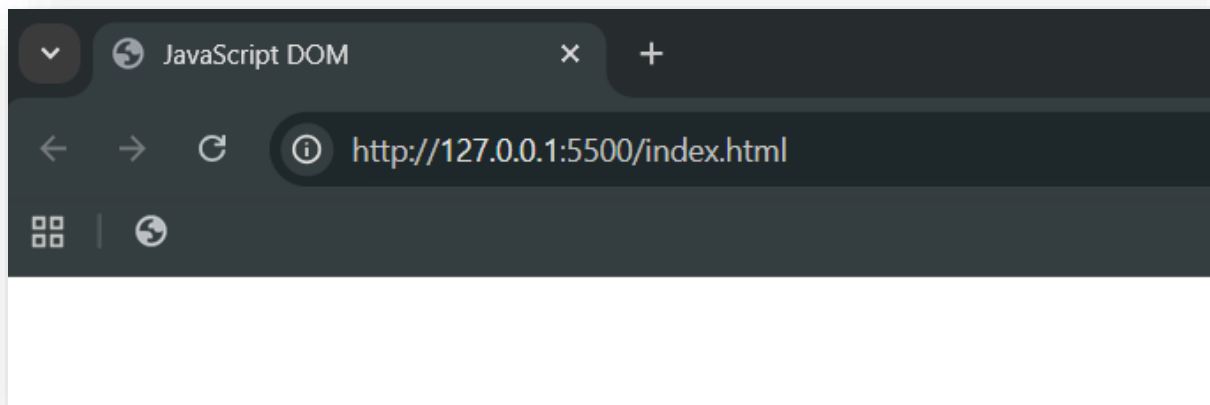
1. It parses the HTML.
2. Creates a DOM tree (each HTML tag becomes a node).
3. JavaScript can then access and manipulate these nodes.

A very basic example without DOM:

Code:

```
index.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>JavaScript DOM</title>
7  </head>
8  <body>
9
10 </body>
11 </html>
```

Output:

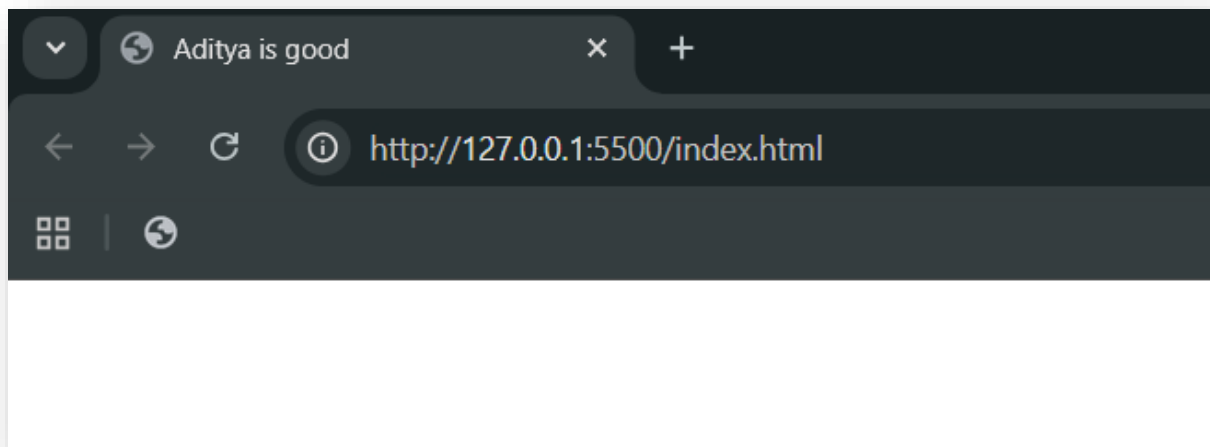


Now, we will change the title of the page using the <script>:

Code:

```
index.html > html > body > script
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>JavaScript DOM</title>
7  </head>
8  <body>
9      <script>
10         //DOM to change the title of the page
11         document.title = "Aditya is good";
12     </script>
13 </body>
14 </html>
```

Output:

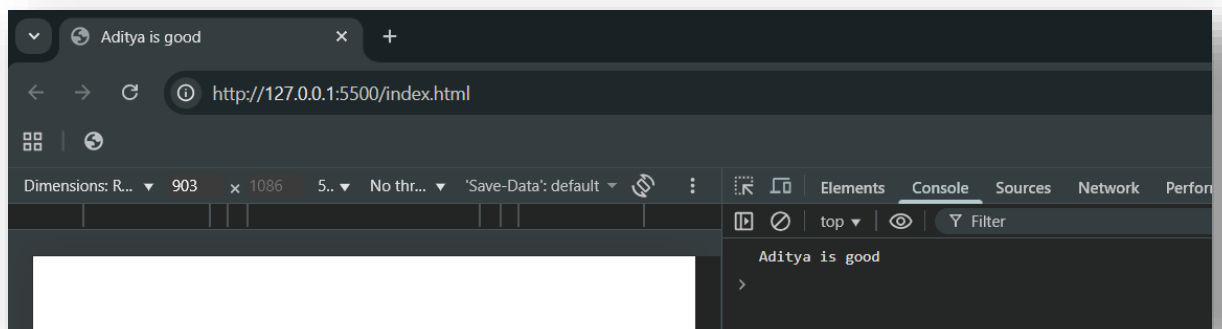


We can also, print the title of the page in the console, using the `console.log()`:

Code:

```
index.html > html > body > script
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |   <meta charset="UTF-8">
5  |   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6  |   <title>JavaScript DOM</title>
7  </head>
8  <body>
9  |   <script>
10 |       //DOM to change the title of the page
11 |       document.title = "Aditya is good";
12 |       console.log(document.title);
13 |   </script>
14 </body>
15 </html>
```

Output:

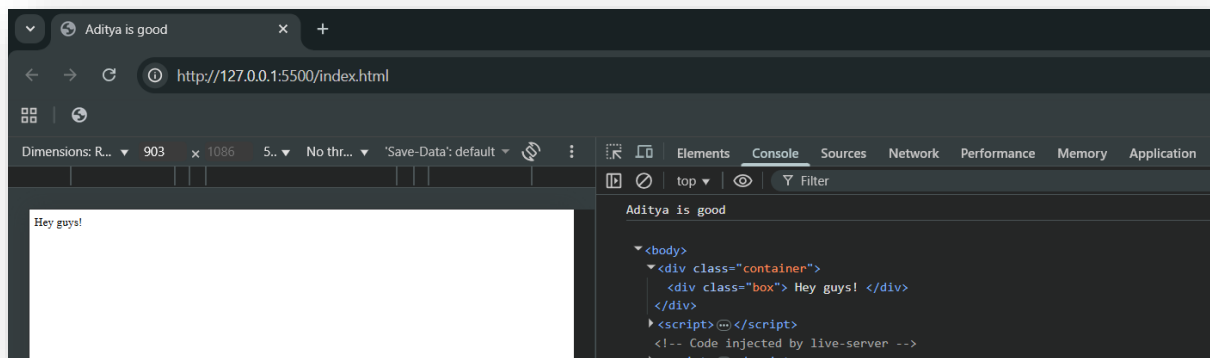


We can also, print the content of the <body>:

Code:

```
index.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>JavaScript DOM</title>
7  </head>
8  <body>
9    <div class="container">
10     <div class="box">
11       Hey guys!
12     </div>
13   </div>
14   <script>
15     //DOM to change the title of the page
16     document.title = "Aditya is good";
17     console.log(document.title);
18     console.log(document.body);
19   </script>
20 </body>
21 </html>
```

Output:

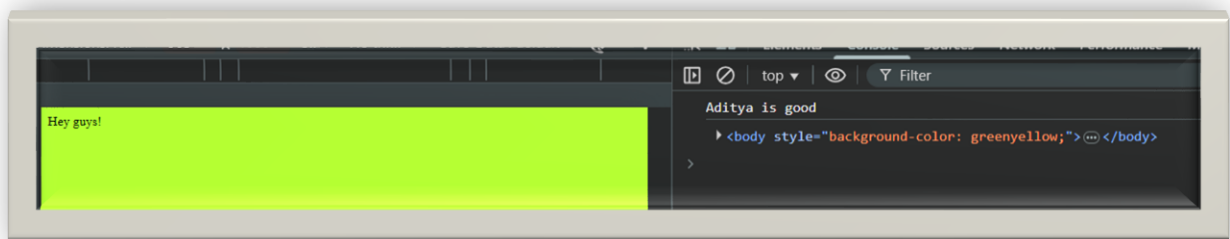


We can also assign properties of CSS through it:

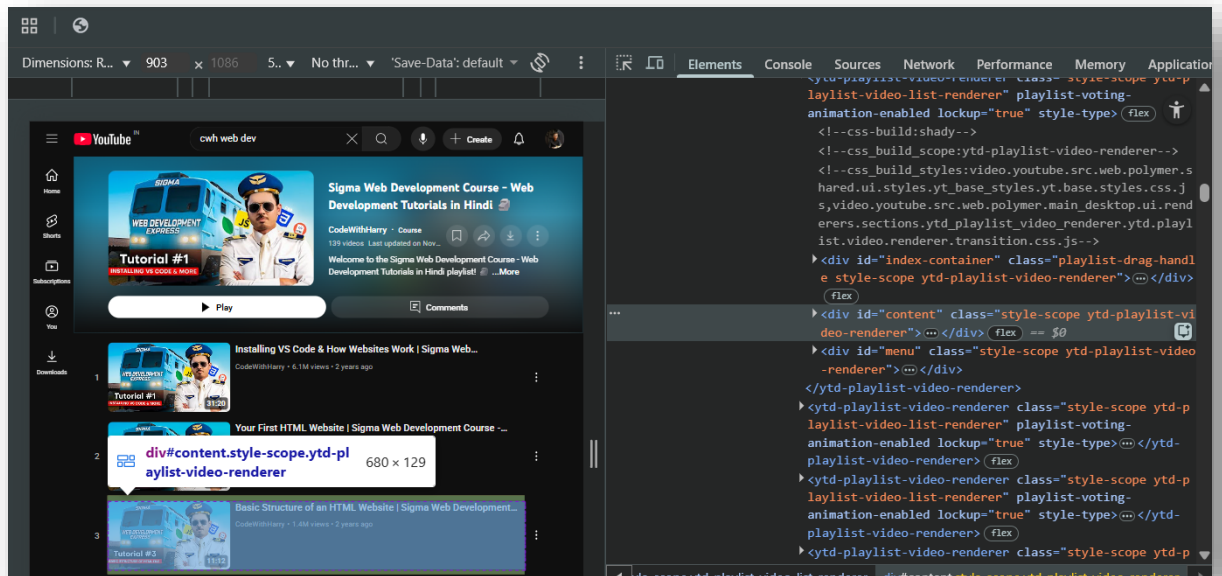
Code:

```
index.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>JavaScript DOM</title>
7  </head>
8  <body>
9      <div class="container">
10         <div class="box">
11             Hey guys!
12         </div>
13     </div>
14     <script>
15         //DOM to change the title of the page
16         document.title = "Aditya is good";
17         console.log(document.title);
18         console.log(document.body);
19         document.body.style.backgroundColor = "greenyellow";
20     </script>
21 </body>
22 </html>
```

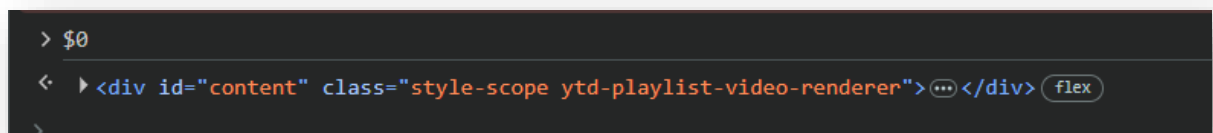
Output:



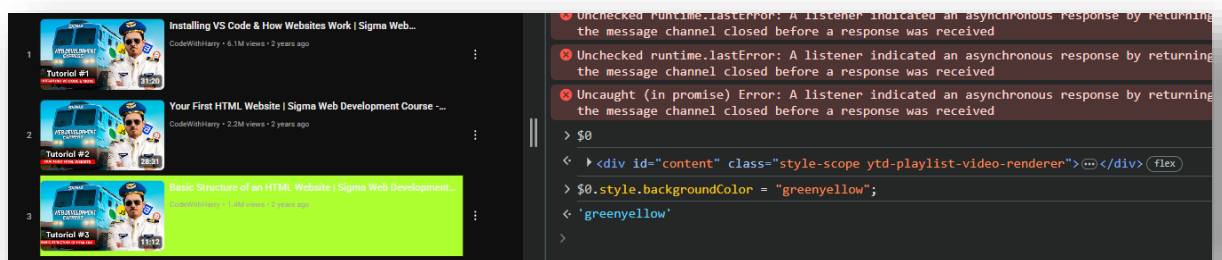
Also, keep in mind that the selected element gets the \$0, so it's easy for us to target it in the browser:



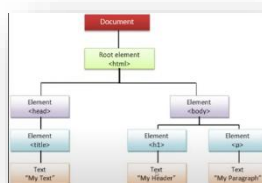
We can even target it: clearly it shows the targeted element



Manipulating using \$0:



DOM:



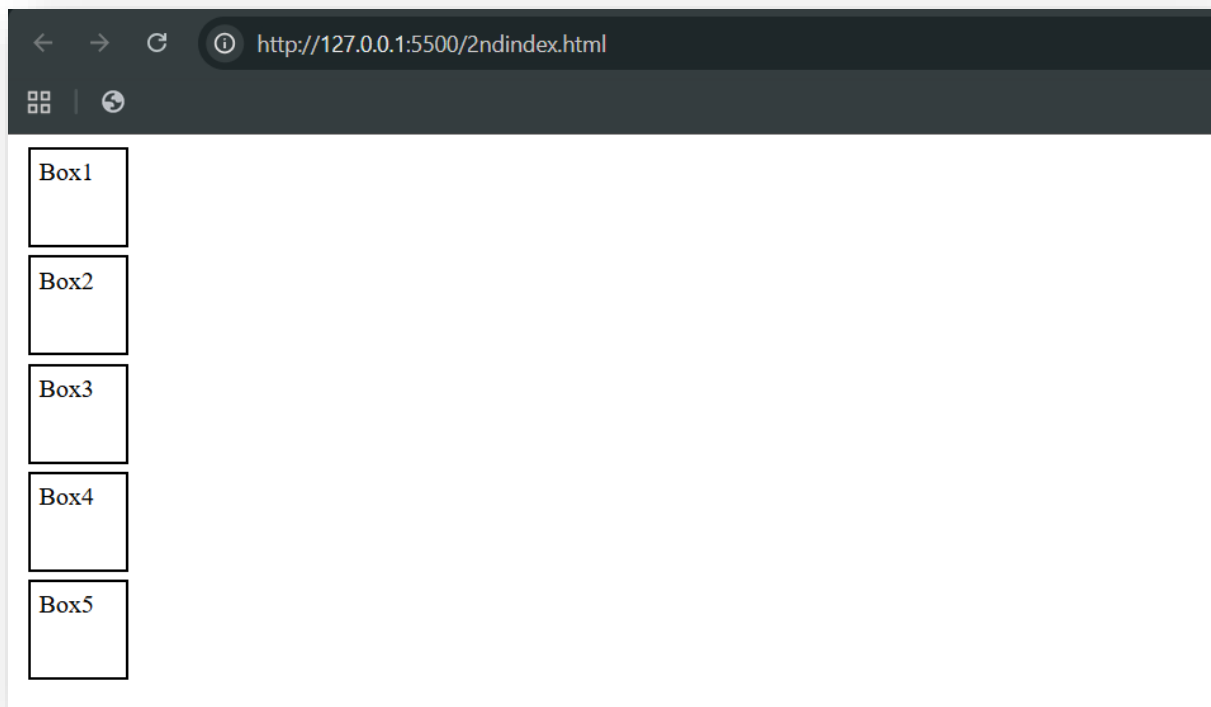
JavaScript DOM - Children, Parent & Sibling Nodes:

Creating a basic page with 5 <div>s:

Code:

```
2ndindex.html > html > head > style > .box
2  <html lang="en">
3  <head>
7      <style>
8          .box{
9              margin: 5px;
10             padding: 5px;
11             border: 2px solid black;
12             height: 50px;
13             width: 50px;
14         }
15     </style>
16 </head>
17 <body>
18     <div class="container">
19         <div class="box">Box1</div>
20         <div class="box">Box2</div>
21         <div class="box">Box3</div>
22         <div class="box">Box4</div>
23         <div class="box">Box5</div>
24     </div>
25 </body>
26 </html>
```

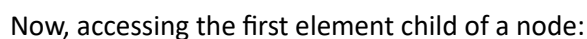
Output:



Suppose we want to know the number of child nodes in the container class: just write it as `document.body.childNodes[1]` as shown below

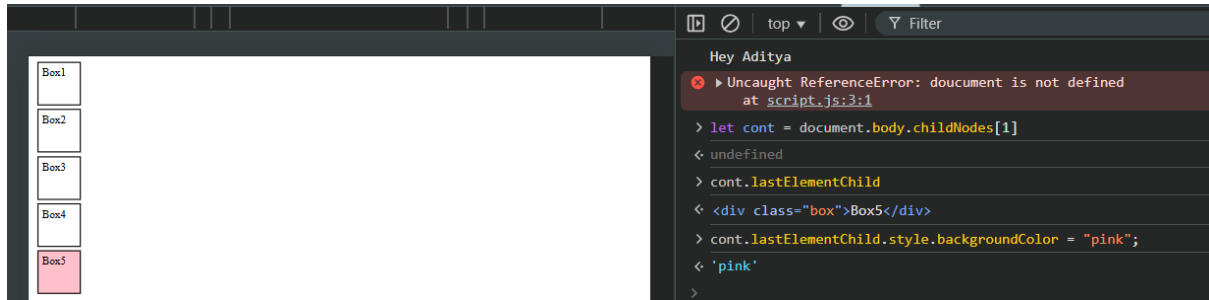


Now, accessing the first child and last child of any node:

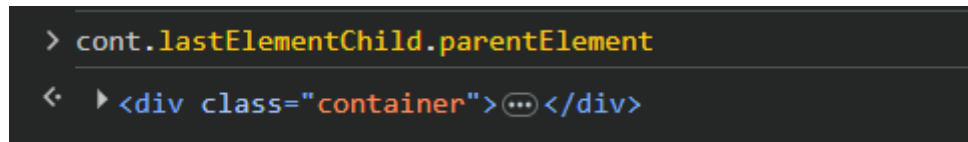


```
> cont.lastElementChild
<div class="box">Box5</div>
```

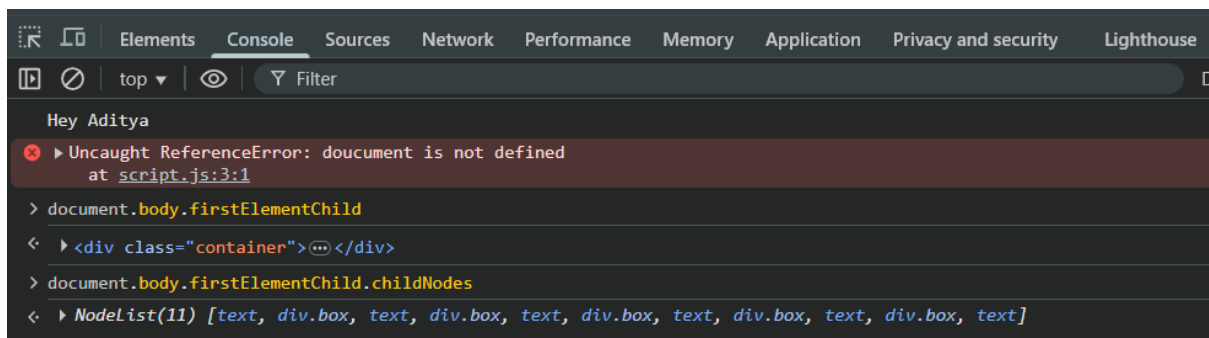

Now, targeting the property of last element child:



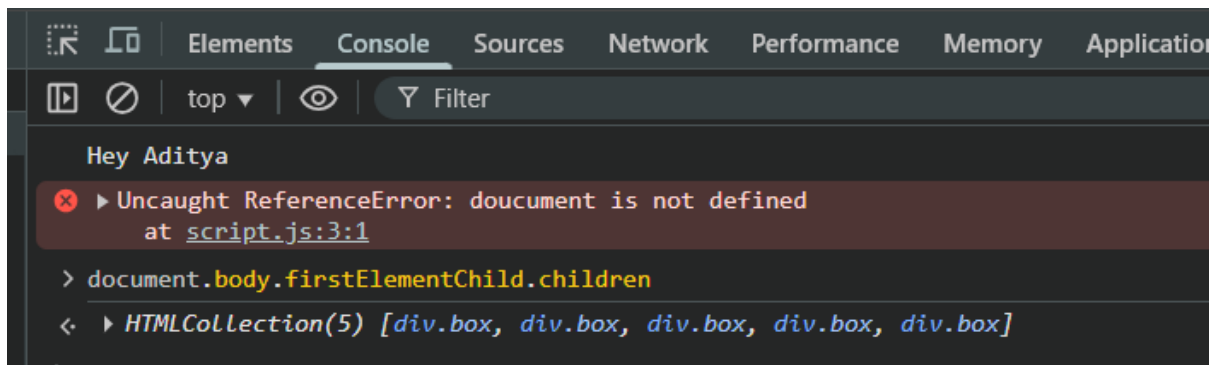
Now, to get the parent node of an element:



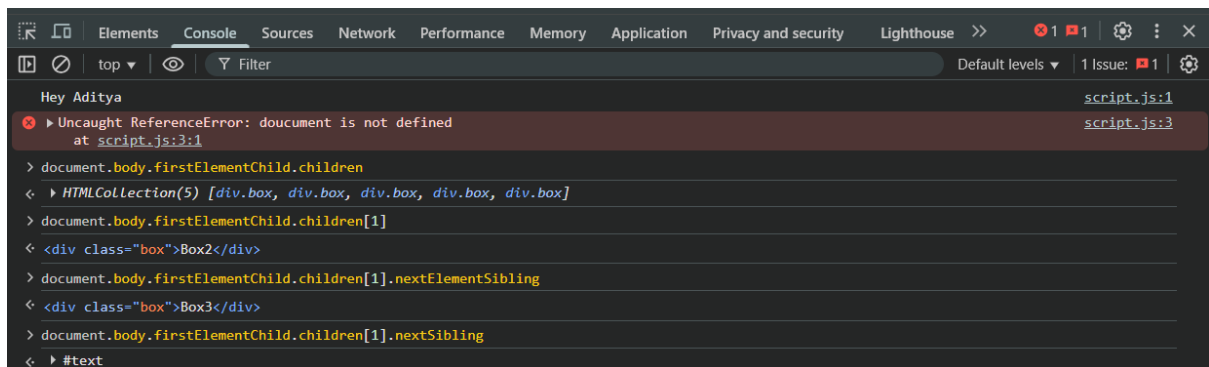
Now, to get all the childnodes of an element child:



In case we just want elements of an element:

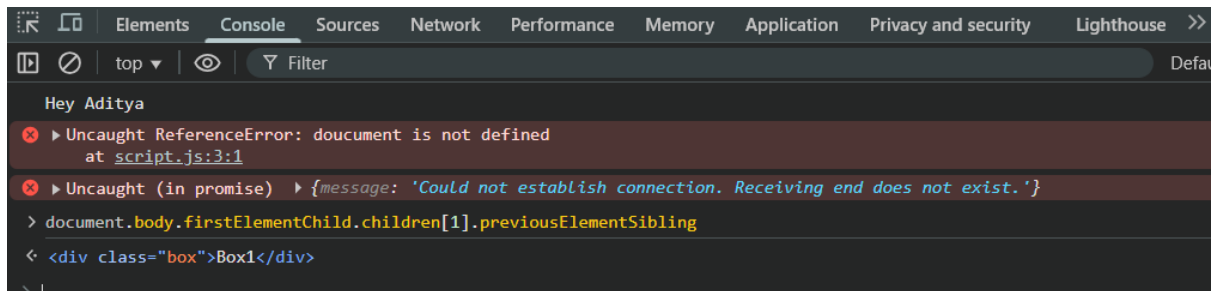


Now, getting the next element sibling as well as just sibling:



A screenshot of the Chrome DevTools Console. The top bar shows tabs for Elements, Console, Sources, Network, Performance, Memory, Application, Privacy and security, and Lighthouse. The Console tab is active, showing a message 'Hey Aditya' and a red error bar: 'Uncaught ReferenceError: document is not defined at script.js:3:1'. Below the error, the following code is executed and its results are shown:
1. `document.body.firstElementChild.children` returns `HTMLCollection(5) [div.box, div.box, div.box, div.box, div.box]`.
2. `document.body.firstElementChild.children[1]` returns `<div class="box">Box2</div>`.
3. `document.body.firstElementChild.children[1].nextElementSibling` returns `<div class="box">Box3</div>`.
4. `document.body.firstElementChild.children[1].nextSibling` returns `#text`.

We can also get the previous element sibling:



A screenshot of the Chrome DevTools Console. The top bar shows tabs for Elements, Console, Sources, Network, Performance, Memory, Application, Privacy and security, and Lighthouse. The Console tab is active, showing a message 'Hey Aditya' and two red error bars: 'Uncaught ReferenceError: document is not defined at script.js:3:1' and 'Uncaught (in promise) {message: 'Could not establish connection. Receiving end does not exist.'}'. Below the errors, the following code is executed and its results are shown:
1. `document.body.firstElementChild.children[1].previousElementSibling` returns `<div class="box">Box1</div>`.
2. The prompt `>` is followed by a blank line.

--The End--