SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING-IOT Including CS&BCT

COURSE NAME : 19SB602 FULL STACK DEVELOPMENT FOR NEXT GENERATION IOT

III YEAR / VI SEMESTER

Unit II- **FRONT-END MODULES**

Topic : Switch, Loops, Functions, HTML DOM

Functions

In Full Stack Development, functions play a vital role in organizing code, promoting reusability, and facilitating modular development.

Definition: Functions are self-contained blocks of code that perform a specific task or calculation.

Abstraction: Functions abstract away implementation details, allowing developers to focus on high-level functionality without needing to understand the internal workings of the function.

Code Reusability: Functions promote code reusability by encapsulating common tasks or operations that can be invoked multiple times across the application.

Modular Development: Functions facilitate modular development by enabling developers to create small, independent units of functionality that can be easily combined to build complex applications.

Scope and Encapsulation: Functions have their own scope, which defines the visibility and accessibility of variables and symbols within the function body.

Higher-Order Functions: Higher-order functions are functions that can accept other functions as arguments or return functions as results.

Asynchronous Programming: Functions are central to asynchronous programming in Full Stack Development, where tasks may execute concurrently or in non-blocking fashion.

HTML DOM

The HTML DOM is an Object Model for HTML.

When a web page is loaded, the browser creates a Document Object Model of the page.

It defines:

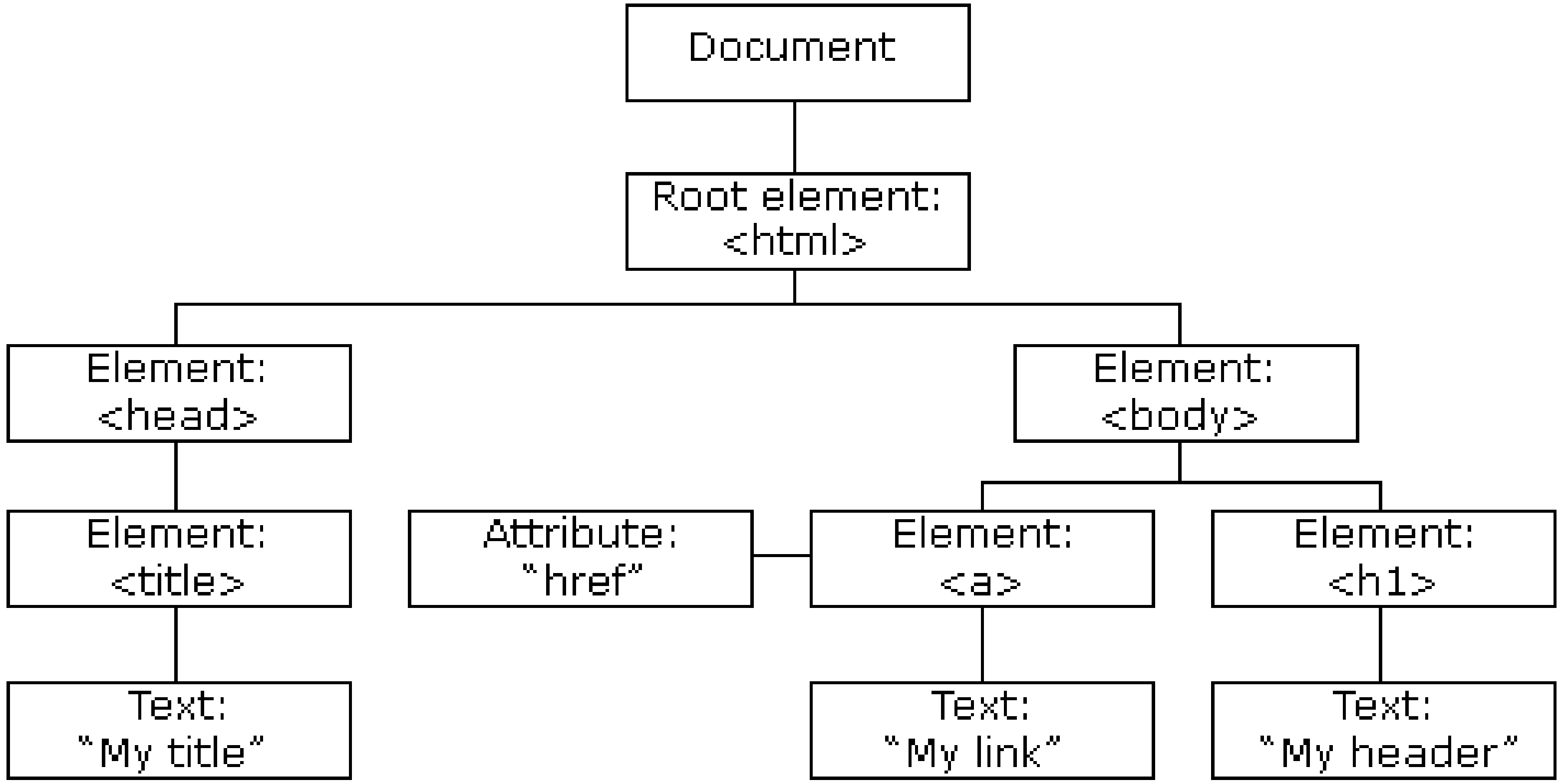
HTML elements as objects
Properties for all HTML elements
Methods for all HTML elements
Events for all HTML elements

The HTML DOM is an API (Programming Interface) for JavaScript:

JavaScript can add/change/remove HTML elements
JavaScript can add/change/remove HTML attributes
JavaScript can add/change/remove CSS styles
JavaScript can react to HTML events
JavaScript can add/change/remove HTML events

The HTML DOM model is constructed as a **tree of Objects**

The HTML DOM Tree of Objects



Finding HTML Elements

When you want to access HTML elements with JavaScript, you have to find the elements first.

There are a couple of ways to do this:

Finding HTML elements by id
Finding HTML elements by tag name
Finding HTML elements by class name
Finding HTML elements by CSS selectors
Finding HTML elements by HTML object collections

Finding HTML Element by Id

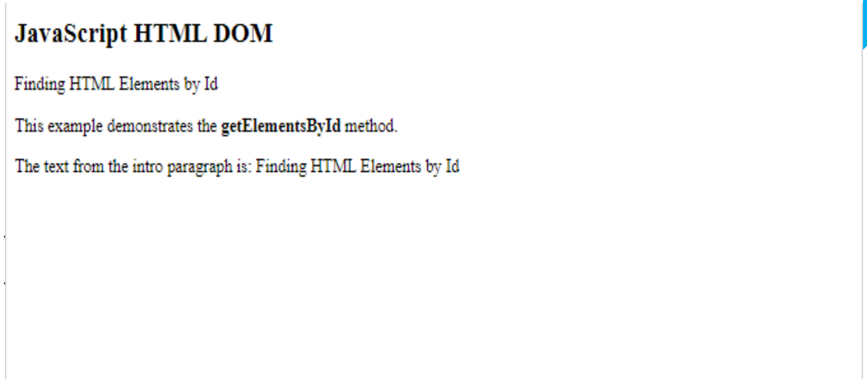
The easiest way to find an HTML element in the DOM, is by using the element id.

This example finds the element with id="intro":

Example

```
var myElement = document.getElementById("intro");
```

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript HTML DOM</h2>

<p id="intro">Finding HTML Elements by Id</p>
<p>This example demonstrates the <b>getElementsById</b> method.</p>

<p id="demo"></p>

<script>
const element = document.getElementById("intro"

document.getElementById("demo").innerHTML =
"The text from the intro paragraph is: " + element.innerHTML;

</script>

</body>
</html>
```



JavaScript HTML DOM

Finding HTML Elements by Id

This example demonstrates the **getElementsById** method.

The text from the intro paragraph is: Finding HTML Elements by Id

If the element is found, the method will return the element as an object (in myElement).

If the element is not found, myElement will contain null.

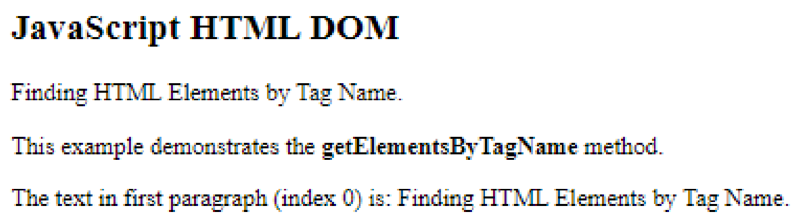
Finding HTML Elements by Tag Name

This example finds all <p> elements:

Example

```
var x = document.getElementsByTagName("p");
```

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript HTML DOM</h2>

<p>Finding HTML Elements by Tag Name.</p>
<p>This example demonstrates the <b>getElementsByTagName</b> method.</p>

<p id="demo"></p>

<script>
const element = document.getElementsByTagName("p");

document.getElementById("demo").innerHTML = 'The text in first paragraph (index 0) is: ' +
element[0].innerHTML;

</script>

</body>
</html>
```



JavaScript HTML DOM

Finding HTML Elements by Tag Name.

This example demonstrates the **getElementsByTagName** method.

The text in first paragraph (index 0) is: Finding HTML Elements by Tag Name.

Finding HTML Elements by HTML Object Collections

HTML object collections are also accessible:

document.anchors
document.forms
document.images
document.links
document.scripts

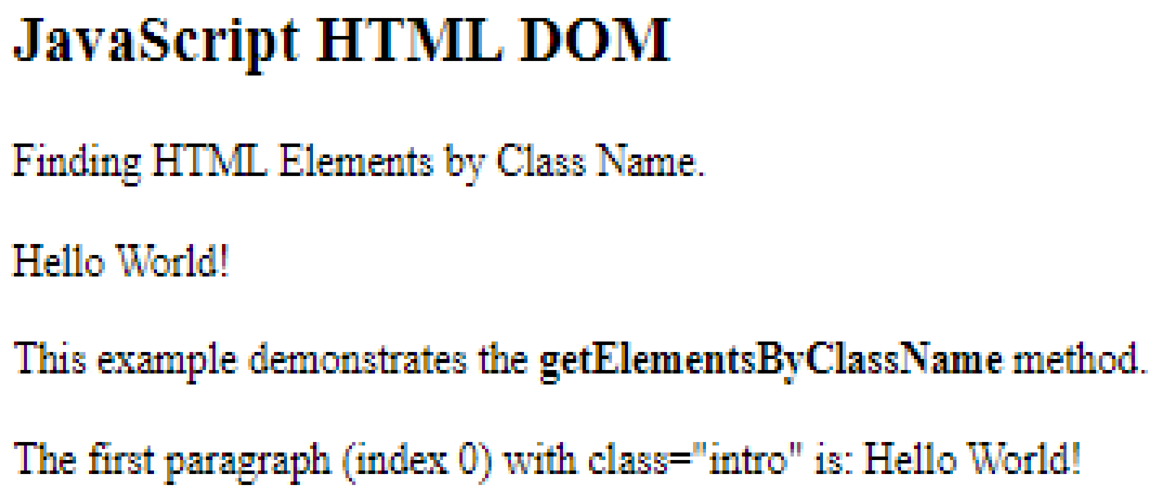
Finding HTML Elements by Class Name

If you want to find all HTML elements with the same class name, use getElementsByClassName().
This example returns a list of all elements with class="intro".

Example

```
var x = document.getElementsByClassName("intro");
```

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript HTML DOM</h2>

<p>Finding HTML Elements by Class Name.</p>
<p class="intro">Hello World!</p>
<p class="intro">This example demonstrates the <b>getElementsByClassName</b>
method.</p>

<p id="demo"></p>

<script>
const x = document.getElementsByClassName("intro");
document.getElementById("demo").innerHTML =
'The first paragraph (index 0) with class="intro" is: ' + x[0].innerHTML;
</script>

</body>
</html>
```



JavaScript HTML DOM

Finding HTML Elements by Class Name.

Hello World!

This example demonstrates the **getElementsByClassName** method.

The first paragraph (index 0) with class="intro" is: Hello World!

Any Query????

Thank you……