function second()

{

  let x = 10;

  console.log(x);

  {  let x= 100;

    console.log(x);

  }

console.log(x);

}

function third()

{

  const x = 10;

  console.log(x);

  {  x= 100;

    console.log(x);

  }

x=100;

console.log(x);

}

function first()

{

  var x = 10;

  console.log(x);

  {   x = 100;

    console.log(x);

  }

console.log(x);

}

function second()

{

  // let x = 10;

  //console.log(x);

  {  let x= 100;

    console.log(x);

  }

console.log(x);

}

function third()

{

  const x = 10;

  console.log(x);

  {  x= 100;

    console.log(x);

  }

x=100;

console.log(x);

}

function first()

{

  // var x = 10;

  // console.log(x);

  {   x = 100;

    console.log(x);

  }

console.log(x);

}

With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

The HTML DOM (Document Object Model)

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

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

The HTML DOM Tree of Objects



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

* JavaScript can change all the HTML elements in the page
* JavaScript can change all the HTML attributes in the page
* JavaScript can change all the CSS styles in the page
* JavaScript can remove existing HTML elements and attributes
* JavaScript can add new HTML elements and attributes
* JavaScript can react to all existing HTML events in the page
* JavaScript can create new HTML events in the page

<!DOCTYPE html>

<html>

  <head>

<script>

  function Call()

  {

    document.getElementById("d1").innerHTML="The contents are changed";

  }

</script>

  </head>

  <body>

    <div id="d1">  Hello  </div>

    <button type="button" onclick="Call()"> Click </button>

  </body>

</html>

<!DOCTYPE html>

<html>

  <head>

    <style>

      /\* selector

      {

      style rule

      } \*/

      /\*  Selector could be of 3 types

      1, Element

      2, class

      3, id  \*/

 /\*  element ex \*/

  div, h1 , p{

    background-color: rgb(155, 155, 153);

    font-family: Verdana, Geneva, Tahoma, sans-serif;

    color: green;

  }

  /\* class example \*/

  .class1

  {

     background-color: red;

     font-size: xx-large;

     border-style: dashed;

     border-width: 20px;

     border-color: greenyellow;

  }

  /\* id example  \*/

   #id1

   {

    color: indigo;

    background-color: pink;

   }

    </style>

<script>

  function Call()

  {

    document.getElementById("d1").innerHTML="The contents are changed";

  }

</script>

  </head>

  <body>

    <h1 id="id1"> Heading One </h1>

    <div id="d1" style="background-color: red;">  Hello  </div>

    <button type="button" onclick="Call()"> Click </button>

    <div class="class1"> DIV 1 </div>

    <p class="class1"> The Storage component of Windows Azure represents a durable store in the cloud. Windows Azure allows developers to store tables, blobs, and message queues. The storage can be accessed through HTTP. You can also create our own client; although Windows Azure SDK provides a client library for accessing the Storage.

      In this chapter, we will learn how to create a Windows Azure Storage account and use it for storing data.

      </p>

  </body>

</html>

What are Semantic Elements in HTML >

What is meant by semantic elements?

Semantic elements = elements with a meaning. ... A semantic element clearly **describes its meaning to both the browser and the developer**. Examples of nonsemantic elements: <div> and <span> Tells nothing about its content. Examples of semantic elements: <form>, <table>, and <article> Clearly defines its content.