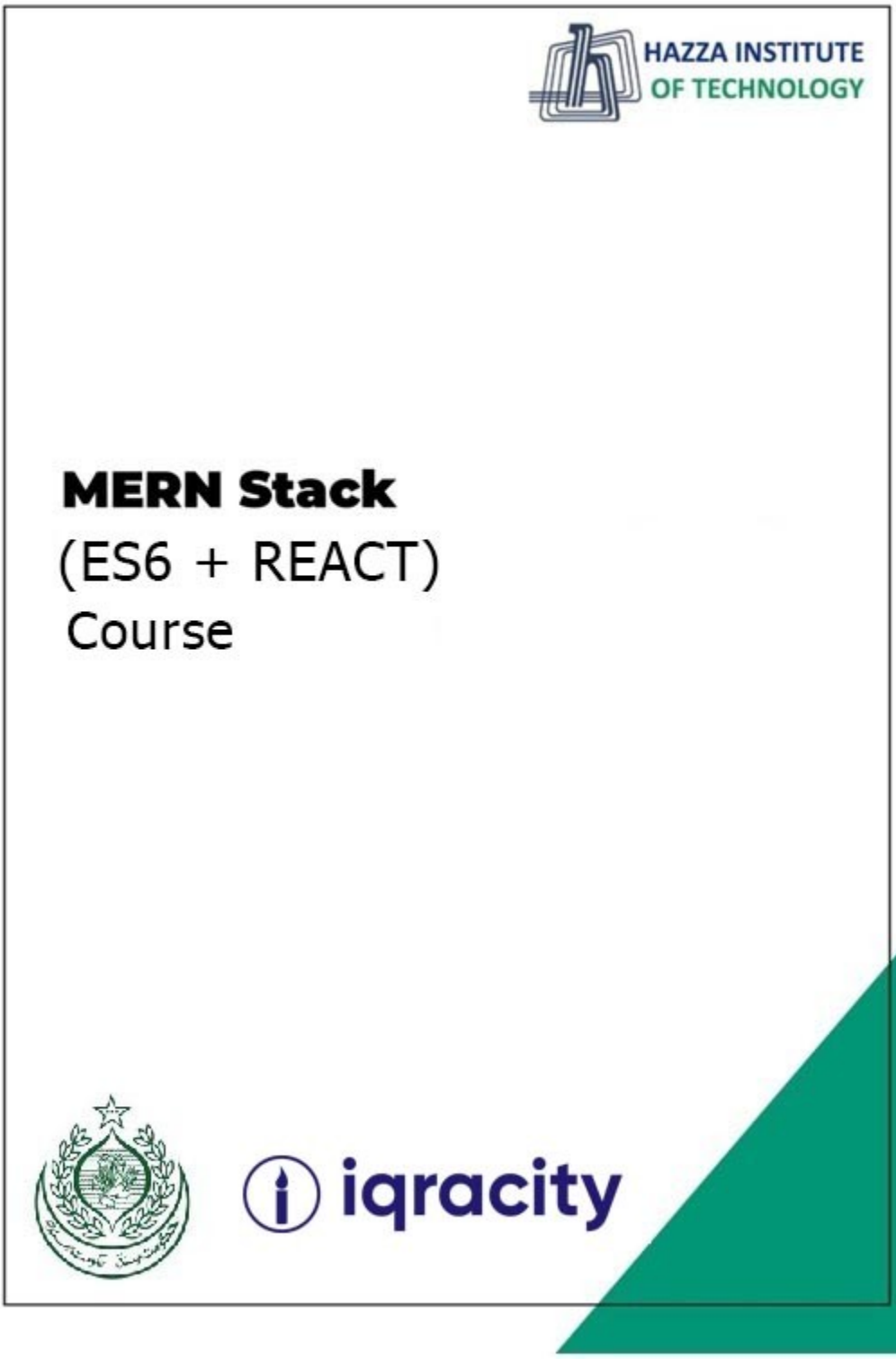


Lab 11

Total Time:
3 hours

Pre-Lab Activities:

- No Pre-Lab Activity

Learning Outcomes:

- Perform the execution, debugging, testing, and profiling of web apps in modern IDEs.

Lab Tasks:

- JS HTML DOM
- DOM Intro
- DOM Methods
- DOM Document
- DOM Elements
- DOM HTML
- DOM Forms

Student Activities:

- To Explore JS HTML DOM
- To Explore DOM Intro
- To Explore DOM Methods
- To Explore DOM Document
- To Explore DOM Elements
- To Explore DOM HTML
- To Explore DOM Forms

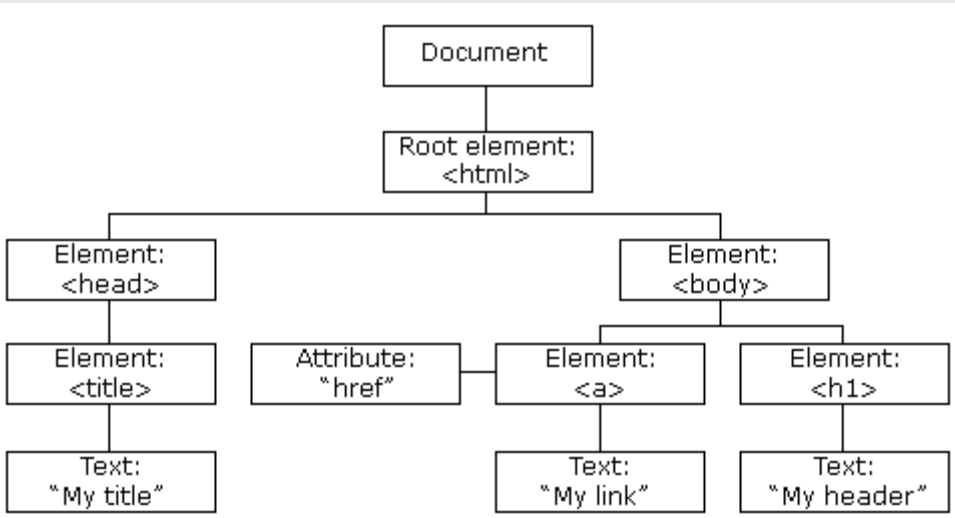
Lab Solution

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

What is the DOM?

The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The W3C DOM standard is separated into 3 different parts:

- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

What is the HTML DOM?

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The **events** for all HTML elements

In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

JavaScript - HTML DOM Methods

HTML DOM methods are **actions** you can perform (on HTML Elements).

HTML DOM properties are **values** (of HTML Elements) that you can set or change.

The DOM Programming Interface

The HTML DOM can be accessed with JavaScript (and with other programming languages).

In the DOM, all HTML elements are defined as **objects**.

The programming interface is the properties and methods of each object.

A property is a value that you can get or set (like changing the content of an HTML element).

A method is an action you can do (like add or deleting an HTML element).

Example

The following example changes the content (the innerHTML) of the <p> element with id="demo":

Example

```
<html>
<body>

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

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

Try it Yourself »

In the example above, getElementById is a **method**, while innerHTML is a **property**.

The getElementById Method

The most common way to access an HTML element is to use the id of the element.

In the example above the getElementById method used id="demo" to find the element.

The innerHTML Property

The easiest way to get the content of an element is by using the innerHTML property.

The innerHTML property is useful for getting or replacing the content of HTML elements.

The innerHTML property can be used to get or change any HTML element, including <html> and <body>.

JavaScript HTML DOM Document

The HTML DOM document object is the owner of all other objects in your web page.

The HTML DOM Document Object

The document object represents your web page.

If you want to access any element in an HTML page, you always start with accessing the document object.

Below are some examples of how you can use the document object to access and manipulate HTML.

Finding HTML Elements

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Changing HTML Elements

Property	Description
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML elem
element.style.*property = new style*	Change the style of an HTML element
Method	Description
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML elem

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element

document.replaceChild(*new, old*)	Replace an HTML element
document.write(*text*)	Write into the HTML output

Adding and Deleting Elements

Adding Events Handlers

Method	Description
document.getElementById(*id*).onclick = function(){*code*}	Adding event handler code to an onclick eve

Finding HTML Objects

The first HTML DOM Level 1 (1998), defined 11 HTML objects, object collections, and properties. These are still valid in HTML5.

Later, in HTML DOM Level 3, more objects, collections, and properties were added.

Property	Description
document.anchors	Returns all <a> elements that have a name attribute

document.applets	Deprecated
document.baseURI	Returns the absolute base URI of the document
document.body	Returns the <body> element
document.cookie	Returns the document's cookie
document.doctype	Returns the document's doctype
document.documentElement	Returns the <html> element
document.documentMode	Returns the mode used by the browser
document.documentURI	Returns the URI of the document
document.domain	Returns the domain name of the document server
document.domConfig	Obsolete.
document.embeds	Returns all <embed> elements

document.forms	Returns all <form> elements
document.head	Returns the <head> element
document.images	Returns all <img> elements
document.implementation	Returns the DOM implementation
document.inputEncoding	Returns the document's encoding (character set)
document.lastModified	Returns the date and time the document was updated
document.links	Returns all <area> and <a> elements that have a href attribute
document.readyState	Returns the (loading) status of the document
document.referrer	Returns the URI of the referrer (the linking document)
document.scripts	Returns all <script> elements
document.strictErrorChecking	Returns if error checking is enforced

document.title	Returns the <title> element
document.URL	Returns the complete URL of the document

JavaScript HTML DOM Elements

This page teaches you how to find and access HTML elements in an HTML page.

Finding HTML Elements

Often, with JavaScript, you want to manipulate HTML elements.

To do so, you have to find the elements first. There are several 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

```
const element = document.getElementById("intro");
```

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

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

Finding HTML Elements by Tag Name

This example finds all <p> elements:

Example

```
const element = document.getElementsByTagName("p");
```

Try it Yourself »

This example finds the element with id="main", and then finds all <p> elements inside "main":

Example

```
const x = document.getElementById("main");
const y = x.getElementsByTagName("p");
```

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

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

Finding HTML Elements by CSS Selectors

If you want to find all HTML elements that match a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the querySelectorAll() method.

This example returns a list of all <p> elements with class="intro".

Example

```
const x = document.querySelectorAll("p.intro");
```

Finding HTML Elements by HTML Object Collections

This example finds the form element with id="frm1", in the forms collection, and displays all element values:

Example

```
const x = document.forms["frm1"];
let text = "";
for (let i = 0; i < x.length; i++) {
  text += x.elements[i].value + "<br>";
}
document.getElementById("demo").innerHTML = text;
```

The following HTML objects (and object collections) are also accessible:

- document.anchors
- document.body
- document.documentElement
- document.embeds
- document.forms
- document.head
- document.images
- document.links
- document.scripts
- document.title

Test Yourself With Exercises

Exercise:

Use the getElementById method to find the <p> element, and change its text to "Hello".

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

<script>
          = "Hello";
</script>
```

JavaScript HTML DOM - Changing HTML

The HTML DOM allows JavaScript to change the content of HTML elements.

Changing HTML Content

The easiest way to modify the content of an HTML element is by using the innerHTML property.

To change the content of an HTML element, use this syntax:

document.getElementById(id).innerHTML = new HTML

This example changes the content of a <p> element:

Example

```
<html>
<body>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML = "New text!";
</script>

</body>
</html>
```

Try it Yourself »

Example explained:

- The HTML document above contains a <p> element with id="p1"
- We use the HTML DOM to get the element with id="p1"
- A JavaScript changes the content (innerHTML) of that element to "New text!"

This example changes the content of an <h1> element:

Example

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

<h1 id="id01">Old Heading</h1>
```

```
<script>
const element = document.getElementById("id01");
element.innerHTML = "New Heading";
</script>

</body>
</html>
```

Try it Yourself »

Example explained:

- The HTML document above contains an <h1> element with id="id01"
- We use the HTML DOM to get the element with id="id01"
- A JavaScript changes the content (innerHTML) of that element to "New Heading"

Changing the Value of an Attribute

To change the value of an HTML attribute, use this syntax:

document.getElementById(id).attribute = new value

This example changes the value of the src attribute of an <img> element:

Example

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

<img id="myImage" src="smiley.gif">

<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>

</body>
</html>
```

Try it Yourself »

Example explained:

- The HTML document above contains an <img> element with id="myImage"
- We use the HTML DOM to get the element with id="myImage"
- A JavaScript changes the src attribute of that element from "smiley.gif" to "landscape.jpg"

Dynamic HTML content

JavaScript can create dynamic HTML content:

Date : Sun May 08 2022 19:04:02 GMT+0500 (Pakistan Standard Time)

Example

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

<script>
document.getElementById("demo").innerHTML = "Date : " + Date(); </script>

</body>
</html>
```

Try it Yourself »

---

document.write()

In JavaScript, document.write() can be used to write directly to the HTML output stream:

Example

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

<p>Bla bla bla</p>

<script>
document.write(Date());
```

```
</script>

<p>Bla bla bla</p>

</body>
</html>
```

Never use document.write() after the document is loaded. It will overwrite the document.

Use HTML DOM to change the value of the image's src attribute.

```
<img id="image" src="smiley.gif">

<script>
document.getElementById("image").src = "pic_mountain.jpg";
</script>
```

JavaScript **Forms**

❮ PreviousNext ❯

JavaScript Form Validation

HTML form validation can be done by JavaScript.

If a form field (fname) is empty, this function alerts a message, and returns false, to prevent the form from being submitted:

JavaScript Example

```
function validateForm() {
  let x = document.forms["myForm"]["fname"].value;
  if (x == "") {
    alert("Name must be filled out");
    return false;
  }
}
```

The function can be called when the form is submitted:

HTML Form Example

```
<form name="myForm" action="/action_page.php" onsubmit="return validateForm()" method="post">
Name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>
```

Try it Yourself »

JavaScript Can Validate Numeric Input

JavaScript is often used to validate numeric input:

Please input a number between 1 and 10

Submit

Try it Yourself »

Automatic HTML Form Validation

HTML form validation can be performed automatically by the browser:

If a form field (fname) is empty, the required attribute prevents this form from being submitted:

HTML Form Example

```
<form action="/action_page.php" method="post">
  <input type="text" name="fname" required>
  <input type="submit" value="Submit">
</form>
```

Try it Yourself »

Automatic HTML form validation does not work in Internet Explorer 9 or earlier.

Data Validation

Data validation is the process of ensuring that user input is clean, correct, and useful.

Typical validation tasks are:

- has the user filled in all required fields?
- has the user entered a valid date?
- has the user entered text in a numeric field?

Most often, the purpose of data validation is to ensure correct user input.

Validation can be defined by many different methods, and deployed in many different ways.

Server side validation is performed by a web server, after input has been sent to the server.

Client side validation is performed by a web browser, before input is sent to a web server.

HTML Constraint Validation

HTML5 introduced a new HTML validation concept called **constraint validation**.

HTML constraint validation is based on:

- Constraint validation **HTML Input Attributes**
- Constraint validation **CSS Pseudo Selectors**
- Constraint validation **DOM Properties and Methods**

Constraint Validation HTML Input Attributes

Attribute	Description
disabled	Specifies that the input element should be disabled
max	Specifies the maximum value of an input element
min	Specifies the minimum value of an input element

pattern	Specifies the value pattern of an input element
required	Specifies that the input field requires an element
type	Specifies the type of an input element

Constraint Validation CSS Pseudo Selectors

Selector	Description
:disabled	Selects input elements with the "disabled" attribute specified
:invalid	Selects input elements with invalid values
:optional	Selects input elements with no "required" attribute specified
:required	Selects input elements with the "required" attribute specified
:valid	Selects input elements with valid values