Unit 4

Q 1) Answer the following in brief.
I) What is DOM?
II) What is the use of typeof operator in JavaScript?
Ans.) I) DOM stands for Document Object Model. It is a programming interface for web documents, which represents the structure of an HTML or XML document as a tree-like model. The DOM allows programmers to interact with the elements of a web page and manipulate their content, structure, and styles using scripting languages like JavaScript.

II) The typeof operator in JavaScript is used to determine the data type of a given value or variable. It returns a string indicating the type of the operand. It can be helpful in scenarios where you need to perform different actions based on the type of data. For example, you can use typeof to check if a variable is a number, string, object, function, or undefined, and then handle it accordingly in your code.

Q 2) Answer the following in detail.
I) Write a JavaScript with DOM to print “Good Day” using IF-ELSE condition.
II) Explain all datatypes used in javascript.
III) Create a javaScript code to validate the student registration form with required fields such as (User name,Email,Password,Contact).
Ans.) I) Here's a JavaScript code using DOM to print "Good Day" based on an IF-ELSE condition:

```javascript
<!DOCTYPE html>
<html>
<head>
  <title>Good Day Example</title>
  <script>
    window.onload = function() {
      var currentDate = new Date();
      var currentHour = currentDate.getHours();

      if (currentHour < 12) {
        document.getElementById("greeting").innerHTML = "Good Morning";
      } else if (currentHour < 18) {
        document.getElementById("greeting").innerHTML = "Good Afternoon";
      } else {
        document.getElementById("greeting").innerHTML = "Good Evening";
      }
    };
```

```
  </script>
</head>
<body>
  <h1 id="greeting"></h1>
</body>
</html>
```

This code will display "Good Morning", "Good Afternoon", or "Good Evening" based on the current time of day.

II) JavaScript has several datatypes:

1. Number: Represents numeric values, including integers and floating-point numbers.
2. String: Represents a sequence of characters enclosed in single quotes (') or double quotes ("").
3. Boolean: Represents a logical value, either true or false.
4. Undefined: Represents a variable that has been declared but has not been assigned a value.
5. Null: Represents the intentional absence of any object value.
6. Object: Represents a collection of key-value pairs or properties.
7. Array: Represents an ordered collection of elements enclosed in square brackets ([]).
8. Function: Represents a reusable block of code that performs a specific task.
9. Symbol: Represents a unique and immutable value, often used as an identifier for object properties.

III) Here's a JavaScript code to validate a student registration form with required fields (User name, Email, Password, Contact):

```javascript
function validateForm() {
  var userName = document.forms["registrationForm"]["userName"].value;
  var email = document.forms["registrationForm"]["email"].value;
  var password = document.forms["registrationForm"]["password"].value;
  var contact = document.forms["registrationForm"]["contact"].value;

  if (userName == "") {
    alert("Please enter a user name");
    return false;
  }

  if (email == "") {
```

```
    alert("Please enter an email");
    return false;
  }

  if (password == "") {
    alert("Please enter a password");
    return false;
  }

  if (contact == "") {
    alert("Please enter a contact number");
    return false;
  }

  return true;
}
```

Q 3) Answer the following in brief.
I) Explain alert(), confirm() and prompt() method of window object.
II) How to access object properties in JavaScript?
Ans.)
I) - `alert()`: The `alert()` method is used to display an alert dialog box with a message and an OK button. It is commonly used to show a notification or provide information to the user.

- `confirm()`: The `confirm()` method displays a dialog box with a message and two buttons: OK and Cancel. It is used to get confirmation from the user. When the user clicks OK, it returns `true`. When the user clicks Cancel, it returns `false`.

- `prompt()`: The `prompt()` method displays a dialog box with a message, an input field for the user to enter a response, and two buttons: OK and Cancel. It is used to get input or information from the user. The value entered by the user is returned as a string. If the user clicks Cancel or leaves the input field empty, it returns `null`.

II) In JavaScript, object properties can be accessed using dot notation or bracket notation.

- Dot notation: It is the most common way to access object properties. You use the dot (`.`) followed by the property name to access the value.

Example:

```javascript
var person = {
  name: "John",
  age: 30,
  address: "123 Main St"
};

console.log(person.name); // Output: John
console.log(person.age); // Output: 30
console.log(person.address); // Output: 123 Main St
```

- Bracket notation: It allows you to access object properties using a string or an expression enclosed in brackets (`[]`).

Example:
```javascript
var person = {
  name: "John",
  age: 30,
  address: "123 Main St"
};

console.log(person["name"]); // Output: John
console.log(person["age"]); // Output: 30
console.log(person["address"]); // Output: 123 Main St

// Bracket notation with dynamic property name
var propertyName = "name";
console.log(person[propertyName]); // Output: John
```

Q 4) Answer the following in detail.
I) Write a JavaScript code which checks the contents entered in a form's text element. If the text entered is in the lower cases convert to upper case.
II) Describe various display posibilities of JavaScript.
III) Explain various ways of finding DOM elements using apropriate example.

Ans.) I) Here's a JavaScript code that checks the contents entered in a form's text element and converts it to uppercase if it is in lowercase:

```javascript
```

```
<!DOCTYPE html>
<html>
<head>
  <title>Convert Text to Uppercase Example</title>
  <script>
    function convertToUppercase() {
      var inputText = document.getElementById("textInput").value;

      if (inputText === inputText.toLowerCase()) {
        document.getElementById("convertedText").innerHTML =
inputText.toUpperCase();
      } else {
        document.getElementById("convertedText").innerHTML = inputText;
      }
    }
  </script>
</head>
<body>
  <input type="text" id="textInput">
  <button onclick="convertToUppercase()">Convert to Uppercase</button>
  <p id="convertedText"></p>
</body>
</html>
```

In this code, when the button is clicked, the `convertToUppercase()` function is called. It retrieves the value entered in the text input element using `document.getElementById()`. It then checks if the input text is already in lowercase by comparing it with its lowercase version (`inputText === inputText.toLowerCase()`). If it is in lowercase, the function sets the content of the `<p>` element with the converted uppercase text using `toUpperCase()`. If it is not in lowercase, it simply displays the original text.

II) JavaScript offers various possibilities for displaying content or interacting with the user interface. Some of the common display possibilities are:

1. `alert()`: It displays a simple alert dialog box with a message and an OK button.

2. `confirm()`: It displays a confirmation dialog box with a message and two buttons: OK and Cancel. It returns `true` if the user clicks OK and `false` if the user clicks Cancel.

3. `prompt()`: It displays a dialog box with a message, an input field for the user to enter a response, and two buttons: OK and Cancel. It returns the value entered by the user as a string.

4. Changing HTML content: JavaScript can manipulate the content of HTML elements by accessing their properties like `innerHTML` or `textContent`. It allows you to dynamically update the text, HTML, or other content within an element.

5. Creating and modifying elements: JavaScript can create new HTML elements using methods like `createElement()` and then append or insert them into the DOM. This allows for dynamic creation and modification of the document structure.

6. Modifying CSS styles: JavaScript can change the style properties of HTML elements by accessing their `style` object. This allows you to modify various visual aspects like color, size, visibility, and positioning.

7. Manipulating classes: JavaScript provides methods like `classList.add()`, `classList.remove()`, and `classList.toggle()` to manipulate CSS classes on elements dynamically. This can be used to change styles or apply different styles based on user interactions.

III) There are various ways to find DOM elements in JavaScript. Here are some common methods along with examples:

1. `getElementById()`: It returns the element with the specified ID attribute.

```javascript
var element = document.getElementById("myElementId");
```

2. `getElementsByClassName()`: It returns a collection of elements with the specified class name.

```javascript
var elements = document.getElementsByClassName("myClass");
```

3. `getElementsByTagName()`: It returns a collection of elements with the specified tag name.

```javascript
var elements = document.getElementsByTagName("div");
```

4. `querySelector()`: It returns the first element that matches the specified CSS selector.

```javascript
var element = document.querySelector("#myElementId");
```

5. `querySelectorAll()`: It returns a collection of elements that

match the specified CSS selector.

```javascript
var elements = document.querySelectorAll(".myClass");
```

Q 5) Answer the following in brief.
I) Define JavaScript.
II) Describe the methods used for adding the script to a webpage.
Ans.) I) JavaScript is a high-level, interpreted programming language that is primarily used for creating interactive and dynamic behavior on webpages. It is a versatile language that runs on the client-side (in the user's web browser) as well as on the server-side (with the help of frameworks like Node.js). JavaScript enables developers to add interactivity, manipulate HTML elements, handle events, perform calculations, make API requests, and much more. It is widely used in web development and has become an essential part of creating modern web applications.

II) There are multiple methods for adding a JavaScript script to a webpage:

1. Inline script: You can include JavaScript directly within the HTML file by using the `<script>` tag and placing the code directly inside it. For example:
```html
<script>
  // JavaScript code here
</script>
```
This method is useful for small snippets of code or quick one-off scripts, but it can make the HTML file less maintainable if the script grows large.

2. External script file: You can create a separate JavaScript file with a `.js` extension and include it in the HTML file using the `<script>` tag with the `src` attribute pointing to the external file. For example:
```html
<script src="script.js"></script>
```

```
This method keeps the HTML file cleaner and separates the JavaScript code into its own file for better organization and reusability.

3. Asynchronous script loading: To improve page loading performance, you can add the `async` attribute to the `<script>` tag. This allows the browser to continue parsing the HTML document while the script is being loaded and executed asynchronously. For example:
```html
<script src="script.js" async></script>
```
Note that the execution order of multiple async scripts may not be guaranteed.

4. Deferred script loading: Similar to asynchronous loading, you can add the `defer` attribute to the `<script>` tag. This defers the script execution until after the HTML document has been parsed, allowing scripts to be executed in the order they appear in the document. For example:
```html
<script src="script.js" defer></script>
```
Deferred scripts will not block the rendering of the page.

5. Dynamic script loading: JavaScript provides methods to dynamically add scripts to a webpage. The `createElement` and `appendChild` methods can be used to create a new `<script>` element and append it to the document. This is useful when you want to load a script based on certain conditions or asynchronously fetch scripts from external sources.

Q 6) Answer the following in detail.
I) Explain alert() dialog box with an example.
II) Explain any five datatypes of JavaScript with an example.
III) Write a JavaScript for following event handlers.
onmouseover()
onclick()
onmouseout()
onfocus()
onblur()
Ans.) I) The `alert()` dialog box is a built-in JavaScript function that displays a modal dialog box with a message and an OK button. It is commonly used to show important information or notifications to the user. When the `alert()` function is called, it interrupts the execution of the script and displays the message as a popup dialog box. The script execution resumes only after the user clicks the OK button.
```

Here's an example of using the `alert()` function:

```javascript
var name = "John";
alert("Hello, " + name + "! Welcome to our website.");
```

In this example, an alert dialog box will be displayed with the message "Hello, John! Welcome to our website." The user needs to click the OK button to dismiss the alert and continue with the script execution.

II) JavaScript has several data types. Here are five common ones with examples:

1. Number: Represents numeric values, including integers and floating-point numbers. Examples:
```javascript
var age = 30;
var price = 9.99;
```

2. String: Represents a sequence of characters enclosed in single quotes ('') or double quotes (""). Examples:
```javascript
var name = "John";
var message = 'Hello, world!';
```

3. Boolean: Represents a logical value, either `true` or `false`. Examples:
```javascript
var isStudent = true;
var isEmployed = false;
```

4. Array: Represents an ordered collection of elements enclosed in square brackets ([]). Examples:
```javascript
var numbers = [1, 2, 3, 4, 5];
var fruits = ["apple", "banana", "orange"];
```

5. Object: Represents a collection of key-value pairs or properties. Examples:
```javascript
var person = {
  name: "John",
```

```
  age: 30,
  city: "New York"
};
var car = {
  brand: "Toyota",
  model: "Camry",
  year: 2021
};
```

III) Here's a JavaScript code that demonstrates event handlers for the following events:

1. `onmouseover()`: Triggered when the mouse pointer enters the element.

```html
<div onmouseover="handleMouseOver()">Mouse over me!</div>

<script>
function handleMouseOver() {
  console.log("Mouse over event occurred");
  // Additional code to handle the event
}
</script>
```

2. `onclick()`: Triggered when the element is clicked.

```html
<button onclick="handleClick()">Click me!</button>

<script>
function handleClick() {
  console.log("Click event occurred");
  // Additional code to handle the event
}
</script>
```

3. `onmouseout()`: Triggered when the mouse pointer leaves the element.

```html
<div onmouseout="handleMouseOut()">Mouse out me!</div>
```

```
<script>
function handleMouseOut() {
  console.log("Mouse out event occurred");
  // Additional code to handle the event
}
</script>
```

4. `onfocus()`: Triggered when the element receives focus (e.g., when it is clicked or selected).

```html
<input type="text" onfocus="handleFocus()" placeholder="Enter your name">

<script>
function handleFocus() {
  console.log("Focus event occurred");
  // Additional code to handle the event
}
</script>
```

5. `onblur()`: Triggered when the element loses focus (e.g., when another element is clicked or selected).

```html
<input type="text" onblur="handleBlur()" placeholder="Enter your name">

<script>
function handleBlur() {
  console.log("Blur event occurred");
  // Additional code to handle the event
}
</script>
```

Q 7) Answer the following in brief.
I) Enlist and explain the methods that can be used with window object. II) Write a JavaScript code to alert “Hello World!”.
Ans.) I) The `window` object in JavaScript represents the current browser window or tab. It provides several methods that can be used for various purposes. Here are some commonly used methods of the `window` object:

1. `alert()`: Displays an alert dialog box with a message.

2. `confirm()`: Displays a confirmation dialog box with OK and Cancel buttons. Returns `true` if the user clicks OK and `false` if the user clicks Cancel.

3. `prompt()`: Displays a dialog box with an input field for the user to enter a response. Returns the value entered by the user as a string.

4. `open()`: Opens a new browser window or tab with the specified URL.

5. `close()`: Closes the current browser window or tab.

6. `setTimeout()`: Executes a function or evaluates an expression after a specified delay in milliseconds.

7. `setInterval()`: Executes a function or evaluates an expression repeatedly at a specified interval in milliseconds.

8. `clearTimeout()`: Cancels a timeout previously set with `setTimeout()`.

9. `clearInterval()`: Cancels an interval previously set with `setInterval()`.

10. `scrollTo()`: Scrolls the window to a specified position.

These are just a few examples of the methods available on the `window` object. The `window` object provides a wide range of functionality for interacting with the browser window and controlling various aspects of the browser's behavior.

II) Here's a JavaScript code that alerts "Hello World!":

```javascript
<script>
  alert("Hello World!");
</script>
```

Q 8) Answer the following in detail.
I) Which methods are used to access the DOM nodes? Explain anyone with an example.
II) Enlist types of JavaScript. Explain any one with an example.
III) List out different event handlers of JavaScript. Explain any two with a script.

Ans.) I) There are several methods that can be used to access DOM nodes (HTML elements) in JavaScript. Some commonly used methods include:

1. `getElementById()`: This method retrieves an element from the DOM using its unique ID attribute. It returns a single element object.

```javascript
var element = document.getElementById("myElementId");
```

2. `getElementsByClassName()`: This method retrieves elements from the DOM based on their class name. It returns a collection (array-like object) of elements.

```javascript
var elements = document.getElementsByClassName("myClass");
```

3. `getElementsByTagName()`: This method retrieves elements from the DOM based on their tag name. It returns a collection of elements.

```javascript
var elements = document.getElementsByTagName("div");
```

4. `querySelector()`: This method retrieves the first element that matches a specified CSS selector. It returns a single element object.

```javascript
var element = document.querySelector("#myElementId");
```

5. `querySelectorAll()`: This method retrieves all elements that match a specified CSS selector. It returns a collection of elements.

```javascript
var elements = document.querySelectorAll(".myClass");
```

These methods provide different ways to select and retrieve DOM elements based on their ID, class, tag name, or CSS selector. Once you have accessed the elements, you can manipulate their content, style, attributes, or other properties using JavaScript.

Here's an example that demonstrates the `getElementById()` method:

```html
<!DOCTYPE html>
<html>
<head>
  <title>Accessing DOM Elements</title>
</head>
<body>
  <div id="myDiv">This is a div element</div>

  <script>
    var element = document.getElementById("myDiv");
    console.log(element.textContent); // Output: "This is a div element"
  </script>
</body>
</html>
```

In this example, the JavaScript code accesses the `div` element with the ID "myDiv" using the `getElementById()` method. It then retrieves the content of the element using the `textContent` property and logs it to the console.

II) JavaScript has several types, including:

1. Primitive types: These include `number`, `string`, `boolean`, `null`, `undefined`, and `symbol`.

Example:

```javascript
var age = 30; // number
var name = "John"; // string
var isStudent = true; // boolean
var person = null; // null
var value = undefined; // undefined
var id = Symbol("id"); // symbol
```

III) JavaScript provides several event handlers that allow you to respond to different events triggered by user interactions or other actions. Some commonly used event handlers include:

1. `onclick`: This event handler is triggered when an element is clicked.

Example:

```html
<button onclick="handleClick()">Click me!</button>

<script>
function handleClick() {
  console.log("Button clicked!");
  // Additional code to handle the event
}
</script>
```
2. `onchange`: This event handler is triggered when the value of an input element or select element is changed.

Example:

```html
<input type="text" onchange="handleChange()" placeholder="Enter your name">

<script>
function handleChange() {
  var input = document.querySelector("input");
  console.log("Input value changed: " + input.value);
  // Additional code to handle the event
}
</script>
```

Q 9) Answer the following in brief.
I) List out the comparison operators in JavaScript.
II) Write JavaScript that takes three numbers as parameters, and returns the sum of that numbers.

Ans.) I) Comparison operators in JavaScript are used to compare two values and return a boolean value (true or false) based on the comparison. Here are the common comparison operators in JavaScript:

1. Equal to (`==`): Compares two values for equality, allowing type coercion if necessary.

2. Not equal to (`!=`): Compares two values for inequality, allowing type coercion if necessary.

3. Strict equal to (`===`): Compares two values for equality without type coercion. It checks both the value and the data type.

4. Strict not equal to (`!==`): Compares two values for inequality without type coercion. It checks both the value and the data type.

5. Greater than (`>`): Checks if the value on the left is greater than the value on the right.

6. Less than (`<`): Checks if the value on the left is less than the value on the right.

7. Greater than or equal to (`>=`): Checks if the value on the left is greater than or equal to the value on the right.

8. Less than or equal to (`<=`): Checks if the value on the left is less than or equal to the value on the right.

These comparison operators are used in conditional statements, loops, and other situations where comparisons need to be made between values.

II) Here's a JavaScript code that takes three numbers as parameters and returns the sum of those numbers:

```javascript
function calculateSum(num1, num2, num3) {
  var sum = num1 + num2 + num3;
  return sum;
}

var result = calculateSum(5, 10, 15);
console.log(result); // Output: 30
```

Q 10) Answer the following in detail.
I) Explain prompt() dialog box with an example.
II) Create the given HTML form with following validation using JavaScript.
i. All fields should be filled. Error should be displayed in alert box.
ii.Password must be 6 characters long.

III) Explain external JavaScript with an example.

Ans.) I) The `prompt()` dialog box is a built-in JavaScript function that displays a modal dialog box with a message, an input field for the user to enter a response, and OK and Cancel buttons. It is commonly used to gather user input or prompt for information.

When the `prompt()` function is called, it interrupts the execution of the script and displays the dialog box. The user can enter a response in the input field and then click OK or Cancel. If the user clicks OK, the `prompt()` function returns the value entered by the user as a string. If the user clicks Cancel, the function returns `null`.

Here's an example of using the `prompt()` function:

```javascript
var name = prompt("Please enter your name:");
if (name !== null) {
  console.log("Hello, " + name + "!");
} else {
  console.log("User cancelled the prompt.");
}
```

II) To create an HTML form with validation using JavaScript, you can use the following code:

```html
<!DOCTYPE html>
<html>
<head>
```

```
  <title>Form Validation</title>
  <script>
    function validateForm() {
      var username = document.forms["myForm"]["username"].value;
      var email = document.forms["myForm"]["email"].value;
      var password = document.forms["myForm"]["password"].value;
      var contact = document.forms["myForm"]["contact"].value;

      if (username === "" || email === "" || password === "" || contact === "") {
        alert("All fields must be filled.");
        return false;
      }

      if (password.length < 6) {
        alert("Password must be at least 6 characters long.");
        return false;
      }

      return true;
    }
  </script>
</head>
<body>
  <form name="myForm" onsubmit="return validateForm()">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username">

    <label for="email">Email:</label>
    <input type="email" id="email" name="email">

    <label for="password">Password:</label>
    <input type="password" id="password" name="password">

    <label for="contact">Contact:</label>
    <input type="text" id="contact" name="contact">

    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

III) External JavaScript refers to a JavaScript file that is separate from the HTML document and is linked using the `<script>` tag. This approach allows you to keep

your JavaScript code separate from your HTML code, promoting code organization and reusability.

To include an external JavaScript file, you can use the following syntax:

```html
<!DOCTYPE html>
<html>
<head>
  <title>External JavaScript Example</title>
  <script src="script.js"></script>
</head>
<body>
  <h1>Hello, World!</h1>
</body>
</html>
```

Here's an example of the "script.js" file:

```javascript
console.log("External JavaScript file loaded!");

function greet(name) {
  console.log("Hello, " + name + "!");
}

greet("John");
```

Q 11) Answer the following in brief.
I) How to give comments in JavaScript? Give an example.
II) State the matching pairs of event handler and its trigger.

A. onload	**1. The user finishes a form and hits the submit button.**
B. onchange	**2. The page finishes loading.**
C. onfocus	**3. The pointer hovers over a link.**
D. hover	**4. A text-entry field is selected and ready for typing**
E. onsubmit	**5. A user changes her name in a form field.**

Ans.) I) In JavaScript, you can give comments to add explanatory notes or to disable certain code lines. There are two ways to add comments in JavaScript:

1. Single-line comments: Use `//` to add a comment that extends until the end of the line.

Example:
```javascript
// This is a single-line comment
var name = "John"; // Assigning a value to the variable
```

2. Multi-line comments: Use `/*` to begin a comment block and `*/` to end it. The comment can span multiple lines.

Example:
```javascript
/*
This is a multi-line comment
It can span multiple lines
*/
var age = 30; // Another comment
```

II) Matching pairs of event handler and trigger:

A. onload - 2. The page finishes loading.

B. onchange - 5. A user changes her name in a form field.

C. onfocus - 4. A text-entry field is selected and ready for typing.

D. hover - 3. The pointer hovers over a link. (Note: The correct event handler for this trigger is `onmouseover` instead of `hover`.)

E. onsubmit - 1. The user finishes a form and hits the submit button.

Q 12) Answer the following in detail.
I) Explain getElementsByClassName() DOM method with example.
II) Explain the confirm() dialog box with an example.
III) Explain embedded JavaScript with an example.
Ans.) I) The `getElementsByClassName()` method is a DOM method in JavaScript that allows you to select and retrieve a collection of elements based on their class

name. It returns a live HTMLCollection or a NodeList, depending on the browser, which represents a collection of elements that have the specified class name.

Here's an example that demonstrates the usage of the `getElementsByClassName()` method:

```html
<!DOCTYPE html>
<html>
<head>
  <title>getElementsByClassName() Example</title>
</head>
<body>
  <div class="container">
    <h1 class="heading">Hello, World!</h1>
    <p class="text">This is a paragraph.</p>
    <p class="text">Another paragraph.</p>
  </div>

  <script>
    var elements = document.getElementsByClassName("text");
    console.log(elements);

    for (var i = 0; i < elements.length; i++) {
      console.log(elements[i].textContent);
    }
  </script>
</body>
</html>
```

II) The `confirm()` dialog box is a built-in JavaScript function that displays a modal dialog box with a message and two buttons: OK and Cancel. It is commonly used to prompt the user for confirmation or to obtain a yes-or-no response.

When the `confirm()` function is called, it interrupts the execution of the script and displays the dialog box. The user can click OK to confirm or Cancel to reject the action. The `confirm()` function returns a boolean value: `true` if the user clicks OK and `false` if the user clicks Cancel.

Here's an example of using the `confirm()` function:

```javascript
var result = confirm("Are you sure you want to delete this item?");
```

```
if (result === true) {
  console.log("Item deleted.");
} else {
  console.log("Deletion cancelled.");
}
```

III) Embedded JavaScript refers to the practice of including JavaScript code within an HTML document directly, typically within the `<script>` tag. The JavaScript code is placed directly in the HTML file and executed by the browser when the document is loaded or when certain events occur.

Here's an example of embedded JavaScript within an HTML document:

```html
<!DOCTYPE html>
<html>
<head>
  <title>Embedded JavaScript Example</title>
</head>
<body>
  <h1>Hello, World!</h1>

  <script>
    var name = "John";
    var age = 30;

    console.log("My name is " + name + " and I am " + age + " years old.");
  </script>
</body>
</html>
```

Q 13) Answer the following in brief.
I) What is the use of popup boxes? Enlist types of popup boxes.
II) Which methods are used to add and remove the DOM nodes?
Ans.) I) Popup boxes are dialog boxes that are displayed on a webpage to provide information, gather user input, or prompt for confirmation. They are used to interact with the user and enhance the user experience. Popup boxes can be used for various purposes such as displaying alerts, confirming actions, collecting user input, or providing additional information.

Types of popup boxes in JavaScript are:

1. Alert box (`alert()`): It displays a simple message box with a specified message and an OK button. It is often used to display important information or notify the user of an event.

2. Confirm box (`confirm()`): It displays a message box with a message and two buttons: OK and Cancel. It is used to prompt the user for confirmation or a yes-or-no response.

3. Prompt box (`prompt()`): It displays a message box with a message, an input field, and two buttons: OK and Cancel. It is used to prompt the user for input and retrieve a value.

II) The methods used to add and remove DOM nodes are:

1. Adding DOM nodes:
   - `createElement(tagName)`: Creates a new DOM element node with the specified tag name.
   - `appendChild(node)`: Appends a child node to an existing parent node.
   - `insertBefore(newNode, referenceNode)`: Inserts a new node before a specified reference node.
   - `innerHTML = htmlContent`: Sets the HTML content of an element, creating and inserting new DOM nodes.

2. Removing DOM nodes:
   - `removeChild(node)`: Removes a child node from its parent node.
   - `parentNode.removeChild(node)`: Removes a specific node from its parent node.
   - `innerHTML = "`: Clears the HTML content of an element, effectively removing its child nodes.

Q 14) Answer the following in detail.
I) Explain three native functions in JavaScript with an example.
II) Enlist methods used for applying event handlers to items within the webpage. Explain anyone of them with an example.
III) Define DOM. Explain JavaScript HTML DOM tree structure.
Ans.) I) Native functions in JavaScript are built-in functions that are provided by the JavaScript language itself. These functions are available for use without the need to define them explicitly. Here are three examples of native functions in JavaScript:

1. `parseInt()`: This function is used to convert a string into an integer. It takes two arguments: the string to be converted and an optional radix (base) specifying the numeral system to be used (default is base 10). It parses the string from left to right until it encounters a character that is not a valid digit in the specified base.

Example:

```javascript
var number = parseInt("42");
console.log(number); // Output: 42

var binaryNumber = parseInt("1010", 2);
console.log(binaryNumber); // Output: 10
```

2. `Math.random()`: This function returns a random floating-point number between 0 (inclusive) and 1 (exclusive). It is commonly used to generate random numbers in JavaScript.

Example:
```javascript
var randomNum = Math.random();
console.log(randomNum); // Output: a random number between 0 and 1
```

3. `Array.isArray()`: This function is used to check if a value is an array. It takes one argument and returns `true` if the value is an array, otherwise returns `false`.

Example:
```javascript
var arr = [1, 2, 3];
console.log(Array.isArray(arr)); // Output: true

var str = "Hello";
console.log(Array.isArray(str)); // Output: false
```

These native functions provide useful functionality in JavaScript and can be used directly in your code without the need for any additional setup or import.

II) There are multiple methods used for applying event handlers to items within a webpage. One of the commonly used methods is the `addEventListener()` method. This method allows you to attach an event handler function to a specified element, enabling you to respond to specific events triggered by that element.

Here's an example of using the `addEventListener()` method to apply an event handler to a button element:

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

```
<head>
  <title>Event Handler Example</title>
</head>
<body>
  <button id="myButton">Click Me</button>

  <script>
    var button = document.getElementById("myButton");

    button.addEventListener("click", function() {
      console.log("Button clicked!");
    });
  </script>
</body>
</html>
```

III) DOM stands for Document Object Model. It is a programming interface for HTML and XML documents, representing the structure and content of a webpage as a tree-like structure of objects. The DOM allows JavaScript to interact with and manipulate the elements of a webpage dynamically.

The JavaScript HTML DOM tree structure consists of various types of nodes, forming a hierarchy:

1. Document Node: It represents the entire HTML document and is the root of the DOM tree.

2. Element Node: It represents an HTML element and is a child of the document node or another element node. Examples of element nodes include `<div>`, `<p>`, `<h1>`, etc.

3. Attribute Node: It represents an attribute of an HTML element. Attribute nodes are associated with element nodes and provide additional information about the element. Examples of attributes include `

id`, `class`, `src`, etc.

4. Text Node: It represents the actual text content within an element. Text nodes are child nodes of element nodes and contain the text within the element. For example, the text "Hello, World!" inside a `<p>` element would be represented by a text node.

5. Comment Node: It represents a comment within an HTML document. Comment nodes are used to add explanatory notes or disable certain code lines.