I/O OPERATIONS IN JAVASCRIPT

Sai Vardhan T

INPUT METHODS IN JS

- PROMPT()
- CONFIRM()

PROMPT

- Displays a dialog box with a message prompting the user to input text. Returns the text entered by the user.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Prompt Example</title>
</head>
<body>
    <h1>Prompt Example</h1>
    <script>
        var name = prompt("Please enter your name:");
        if (name !== null && name !== "") {
            alert("Hello, " + name + "! Welcome to our website.");
        } else {
            alert("Hello! Welcome to our website.");
        }
    </script>
</body>
</html>
```

CONFIRM

- Displays a dialog box with a message and OK/Cancel buttons. Returns true if the user clicks OK, and false if they click Cancel.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Confirm Example</title>
</head>
<body>
    <h1>Confirm Example</h1>
    <script>
            var isConfirmed = confirm("Are you sure you want to perform this action?");
            if (isConfirmed) {
                alert("Action performed successfully!");
            } else {
                alert("Action canceled by the user.");
            }
    </script>
</body>
</html>
```

OUTPUT METHODS

- CONSOLE.LOG()
- DOCUMENT WRITE
- INNER HTML
- CONSOLE WARNING
- ALERT
- CONSOLE ERROR

CONSOLE

- Outputs a message to the browser console. It's primarily used for debugging purposes.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>console.log() Example</title>
</head>
<body>
    <h1>console.log() Example</h1>
    <script>
        console.log("This is a message logged to the console.");
        var x = 5;
        var y = 10;
        console.log("The value of x is:", x);
        console.log("The value of y is:", y);
        console.log("The sum of x and y is:", x + y);
    </script>
</body>
</html>
```

DOCUMENT WRITE

- Writes HTML content directly to the document. It will add the data based on the flow of content in HTML document.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>document.write() Example</title>
</head>
<body>
    <h1>document.write() Example</h1>

    <script>
        var a = "Hello";
        var b = "World";
        document.write(a + " " + b);
    </script>
</body>
</html>
```

INNER HTML

- Manipulating the innerHTML property of an HTML element allows you to dynamically update its content.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>innerHTML Example</title>
</head>
<body>
    <h1 id="demo">Original Content</h1>
    <button onclick="changeContent()">Change Content</button>
    <script>
        function changeContent() {
            var element = document.getElementById("demo");
            element.innerHTML = "New Content";
        }
    </script>
</body>
</html>
```

CONSOLE WARNING

- Outputs a warning message to the browser console. It's used to log warnings and potential issues.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>console.warn() Example</title>
</head>
<body>
    <h1>console.warn() Example</h1>

    <script>
        console.warn("This is a warning message!");
    </script>
</body>
</html>
```

ALERT

- Displays a dialog box with a message and an OK button. It's used to display information to the user.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Alert Example</title>
</head>
<body>
    <h1>Alert Example</h1>
    <script>
        alert("Welcome to our website! Click OK to continue browsing.");
    </script>
</body>
</html>
```

CONSOLE ERROR

- Outputs an error message to the browser console. It's used to log errors and issues.

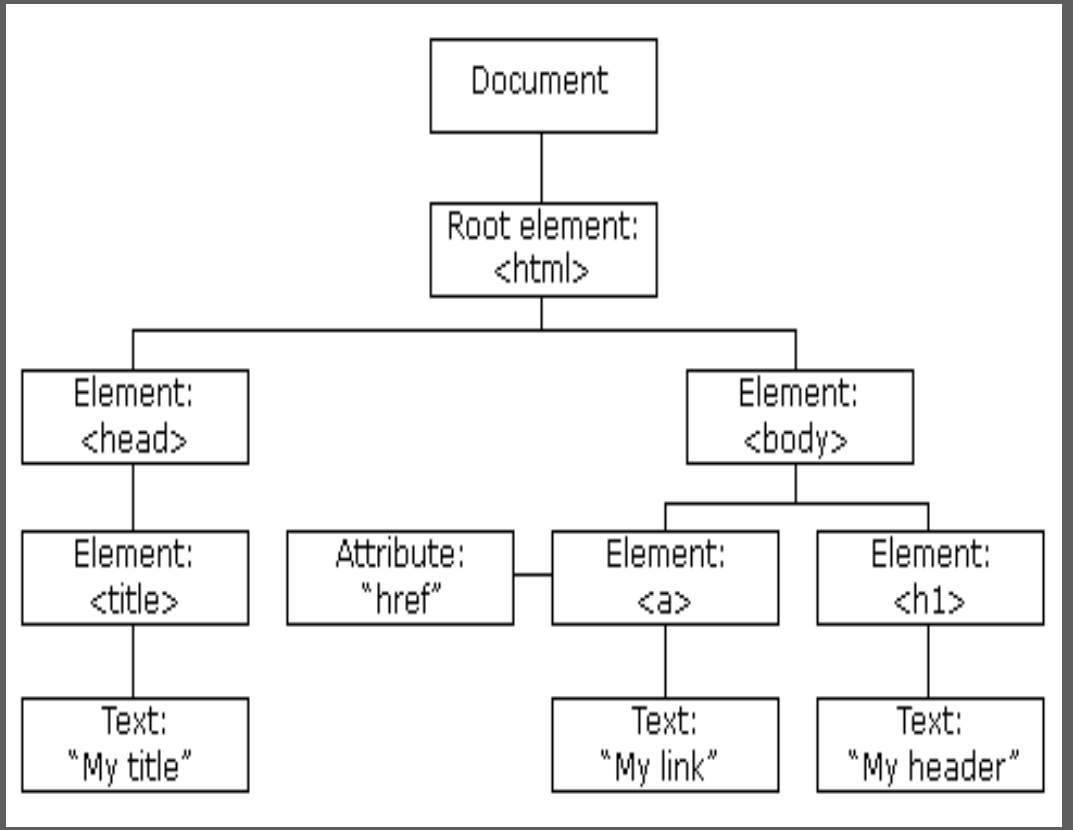
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Console Error Example</title>
</head>
<body>
    <h1>Console Error Example</h1>
    <script>
        // Simulate an error
        var x = 10;
        var y = 2;
        if (y == 0){
            console.error("Zero Division Error Occured");
        }
        else{
            console.log(x/y);
        }
    </script>
</body>
</html>
```

DOM IN JAVASCRIPT

- Document Object Model (DOM) is a programming interface for HTML and XML documents.
- It represents the page so that programs can change the document's structure, style, and content.
- The DOM is an object-oriented representation of a web page, which means that everything on a web page is represented as an object.

DOM IN JAVASCRIPT

```
<html>
    <head>
        <title>My Title</title>
    </head>
    <body>
        <a href="www.google.com">My Link</a>
        <h1>My header</h1>
</html>
```



GETTING VALUES FROM HTML TO JS (FORMS)

- Document.getelementbyid()
- Document.queryselector()
- Document.format()
- Using this
- Using formdata object

Document.getElementById()

- You can retrieve values from form elements by accessing their value property directly or by using the document.getElementById() method to get a reference to the form element.

```
<input type="text" id="myInput">
<button onclick="getValue()">Get Value</button>

<script>
    function getValue() {
        var inputElement = document.getElementById('myInput');
        var value = inputElement.value;
        console.log(value);
    }
</script>
```

document.querySelector()

- Similar to getElementById(), but you can use CSS selectors to retrieve elements.

```
<input type="text" id="myInput">
<button onclick="getValue()">Get Value</button>

<script>
    function getValue() {
        var inputElement = document.querySelector('#myInput');
        var value = inputElement.value;
        console.log(value);
    }
</script>
```

document.forms

- If your form has a name attribute, you can use document.forms to access the form elements.

```
<form name="myForm">
    <input type="text" name="username">
    <button onclick="getValue()">Get Value</button>
</form>

<script>
    function getValue() {
        var inputElement = document.forms['myForm']['username'];
        var value = inputElement.value;
        console.log(value);
    }
</script>
```

Using this

- If you're handling events inline, you can use this to refer to the current element.

```
<input type="text" onchange="getValue(this)">
<script>
    function getValue(element) {
        var value = element.value;
        console.log(value);
    }
</script>
```

Using FormData object

- Forms submitted via AJAX, you can use the FormData object to collect all form field values.

```
<form id="myForm">
    <input type="text" name="username">
    <input type="email" name="email">
    <button onclick="getValue()">Submit</button>
</form>

<script>
    function getValue() {
        var formElement = document.getElementById('myForm');
        var formData = new FormData(formElement);
        for (var pair of formData.entries()) {
            console.log(pair[0] + ': ' + pair[1]);
        }
    }
</script>
```

GETTING VALUES FROM JS TO HTML(FORMS)

- Using innerHTML
- Using textContent
- Using attributes
- Using appendchild()
- Using template literals(ES6)

innerHTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Addition using JavaScript</title>
</head>
<body>
    <p id="result">
    <button onclick="getResult()">Get Result</button><br><br>
    <script>
    function getResult() {
        var x = "I am in Function";
        document.getElementById("result").innerHTML = x;
    }
</script>
</body>
</html>
```

Using textContent

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Using textContent</title>
</head>
<body>
    <h1 id="myElement">
    <button onclick="updateText()">Update Text Content</button>

    <script>
        function updateText() {
            var a = 20;
            var b = 30;
            var c = a+b;
            document.getElementById("myElement").textContent = c;
        }
    </script>
</body>
</html>
```

Using attributes

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Using attributes</title>
</head>
<body>
    <input type="text" id="myInput" value="Original Value">
    <button onclick="updateValue()">Update Value</button>
    <script>
        function updateValue() {
            var x = "Default Value"
            document.getElementById("myInput").value = x;
        }
    </script>
</body>
</html>
```

Using appendchild()

- This will be useful when we want to add external data to existing data.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Using appendChild()</title>
</head>
<body>
    <div id="container">
        <p>Existing Content</p>
    </div>
    <button onclick="addNewElement()">Add New Element</button>
    <script>
        function addNewElement() {
            var newElement = document.createElement("p");
            newElement.textContent = "New Element";
            document.getElementById("container").appendChild(newElement);
        }
    </script>
</body>
</html>
```

Using template literals(ES6)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Using template literals (ES6)</title>
</head>
<body>
    <div id="myElement">Original Content</div>
    <button onclick="updateContent()">Update Content</button>

    <script>
        function updateContent() {
            var data = "Dynamic Data";
            document.getElementById("myElement").innerHTML = `<p>${data}</p>`;
        }
    </script>
</body>
</html>
```