An Introduction to JavaScript JavaScript is a powerful, lightweight, and widely-used scripting language designed for creating interactive web pages. It runs on the client-side within web browsers, allowing dynamic manipulation of HTML and CSS. JavaScript follows an event-driven, functional, and object-oriented programming paradigm. It enables form validation, animations, and interactive user experiences.

Example:

```
console.log("Hello, JavaScript!");
```

Output:

Hello, JavaScript!

JavaScript DOM Model The Document Object Model (DOM) represents the structure of a web page. JavaScript interacts with the DOM to manipulate HTML elements dynamically. The DOM is structured as a tree, where each node represents an element, attribute, or text content. Key DOM methods include:

- `document.getElementById(id)`: Access an element by its ID.
- `document.querySelector(selector)`: Selects an element using CSS selectors.
- `element.innerHTML`: Modifies the content inside an element.
- `element.style.property`: Changes the CSS styles dynamically.

Example:

```
document.getElementById("demo").innerHTML = "Hello, DOM!";
```

Output: Changes the text inside an element with id `demo` to "Hello, DOM!"

Exception Handling JavaScript provides error-handling mechanisms to prevent script failures using `try`, `catch`, `finally`, and `throw` statements.

- `try`: Defines a block of code to test for errors.
- `catch`: Handles the error if an exception occurs.
- `finally`: Executes code after try/catch, regardless of the outcome.
- `throw`: Allows manual error generation.

Example:

```
try {
    let num = 10 / 0;
    if (!isFinite(num)) throw "Divide by zero error";
} catch (error) {
```

```
    console.log("Error:", error);
} finally {
    console.log("Execution completed");
}
```

Output:

```
Error: Divide by zero error
Execution completed
```

Validation in JavaScript Form validation ensures user input is correct before submitting data to the server. JavaScript provides validation using:

- `onSubmit`: Validates input before form submission.
- `onBlur`: Validates input when an element loses focus.
- Regular expressions (`RegExp`) for pattern matching.

Example:

```
function validateForm() {
    let email = document.getElementById("email").value;
    let pattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/;
    if (!pattern.test(email)) {
        alert("Invalid Email Address");
        return false;
    }
    return true;
}
```

Built-in Objects in JavaScript JavaScript has several built-in objects that provide utility functions:

- `Math`: Provides mathematical functions like `Math.random()`, `Math.floor()`.
- `Date`: Handles date and time (`new Date()`, `getFullYear()`).
- `Array`: Allows operations on collections (`push()`, `pop()`, `map()`).
- `String`: Manipulates text (`toUpperCase()`, `split()`).

Example:

```
let today = new Date();
console.log(today.toDateString());
```

Output:

```
Thu Feb 21 2025
```

Event Handling in JavaScript Events allow interaction with users by triggering JavaScript functions. Common events include:

- `onclick`: Executes when an element is clicked.
- `onmouseover`: Executes when the mouse is over an element.
- `onkeydown`: Fires when a key is pressed.

Example:

```
document.getElementById("btn").onclick = function() {
   alert("Button Clicked!");
};
```

Output: Displays an alert box with "Button Clicked!" when the button is clicked.

DHTML with JavaScript Dynamic HTML (DHTML) combines JavaScript, HTML, CSS, and the DOM to create interactive and animated web pages. Examples include:

- Changing styles dynamically (`element.style.color = 'red'`).
- Animating elements (`setTimeout()`, `setInterval()`).
- Hiding and showing elements (`display: none/block`).

Example:

```
document.getElementById("text").style.color = "blue";
```

Output: Changes the text color to blue.

JSON Introduction and Syntax JavaScript Object Notation (JSON) is a lightweight data format used for exchanging data between a server and a client. It is text-based and easy to parse.

Example JSON object:

```
{
   "name": "Alice",
   "age": 25,
   "city": "New York"
}
```

JavaScript parses JSON using:

- `JSON.stringify(obj)`: Converts an object to a JSON string.
- `JSON.parse(string)`: Converts a JSON string into an object.

Example:

```
let jsonData = '{"name": "Alice", "age": 25}';
let obj = JSON.parse(jsonData);
console.log(obj.name);
```

Output:

Alice

Functions in JavaScript Functions allow code reusability and modular programming. They can be:

- Named functions:

```
function add(a, b) {
   return a + b;
}
console.log(add(5, 3));
```

Output:

8

- Anonymous functions:

```
let sum = function(a, b) {
   return a + b;
};
console.log(sum(4, 6));
```

Output:

10

- Arrow functions:

```
const multiply = (a, b) => a * b;
console.log(multiply(3, 4));
```

Output:

12

JavaScript Files JavaScript code can be included in HTML as:

1. Inline: Inside `<script>` tags.
2. Internal: Written inside an HTML file.
3. External: Stored in a `.js` file and linked using:

<script src="script.js"></script>

This method keeps the code modular and enhances performance.