Name: Deher Zainab

Sap ID: 49710

BSCS-5th Semester

Web Assignment # 3

Submitted to:

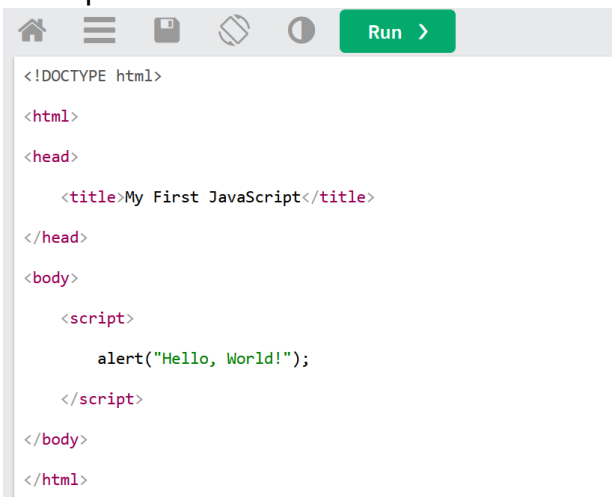
Mam Habiba

JavaScript

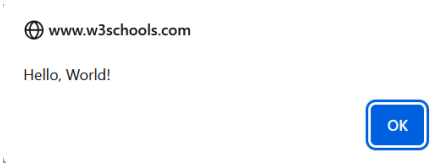
1. **JS Introduction**: JavaScript is a versatile, lightweight programming language used primarily for web development.

2. **JS Where To**: JavaScript can be placed in <script> tags in HTML or linked externally using the src attribute.

Example:



Output: A pop-up alert saying "Hello, World!":



3. **JS Output**: Output can be displayed using `console.log()`, `alert()` or `document.write()` method.

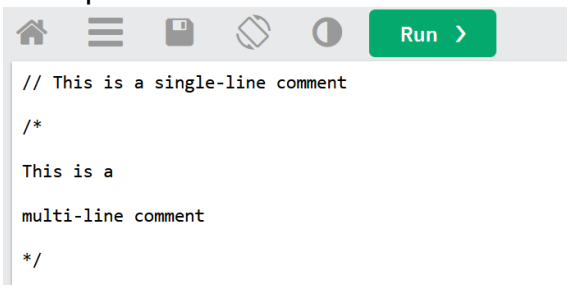
4. **JS Statements**: A statement is a single line of code that performs an action

Example:

5. **JS Syntax**: JavaScript syntax refers to the rules that define a correctly structured program, such as proper use of semicolons and parentheses.

6. **JS Comments**: Comments are added using // for single-line or /* */ for multi-line.

Example:



7. **JS Variables**: Variables store values and can be declared using var, let, or const.

Example:

8. **JS Let**: let is used to declare block-scoped variables.

Example:

9. **JS Const**: const defines a constant variable that cannot be reassigned.

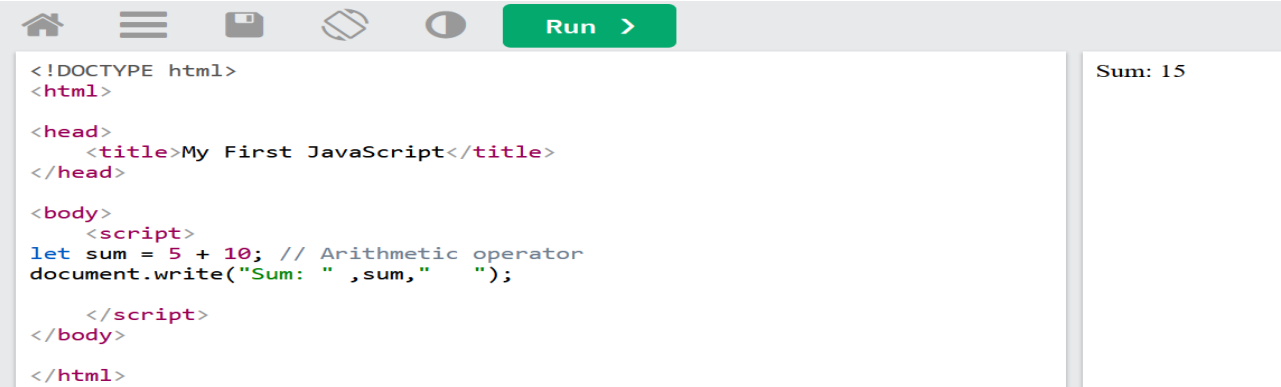
Example:

10. **JS Operators**: Operators perform operations on variables or values, like arithmetic and assignment

    - **JS Arithmetic**: Arithmetic operators perform mathematical operations, such as `+`, `-`, `*`, `/`.

    - **JS Assignment**: Assignment operators like = assign values to variables.

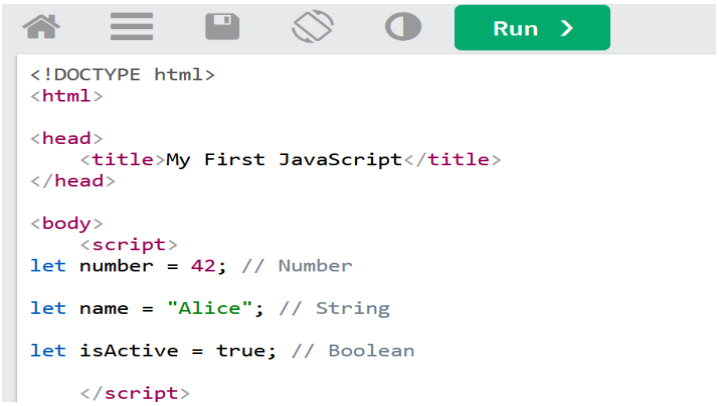
Example 1:



Example 2:

11. **JS Data Types**: JavaScript has several data types, including `String`, `Number`, `Boolean`, `Object`, and `Array`.

Example:

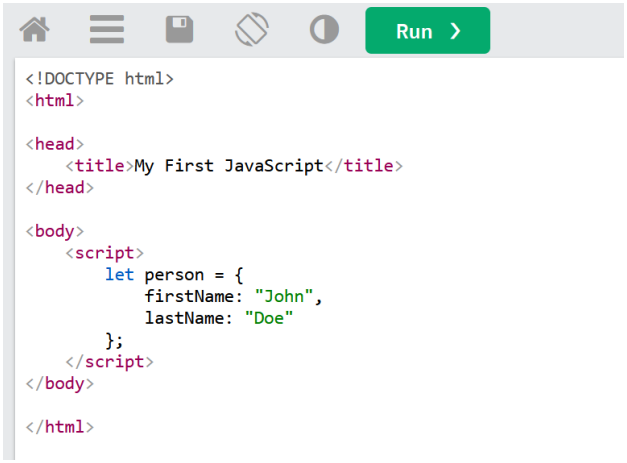


12. **JS Functions**: Functions are blocks of code that perform specific tasks, defined with `function` or arrow syntax.

Example:

13. **JS Objects**: Objects are collections of key-value pairs.

Example:



14. **JS Object Properties**: Object properties are accessed using dot notation or bracket notation, like `obj.name`.

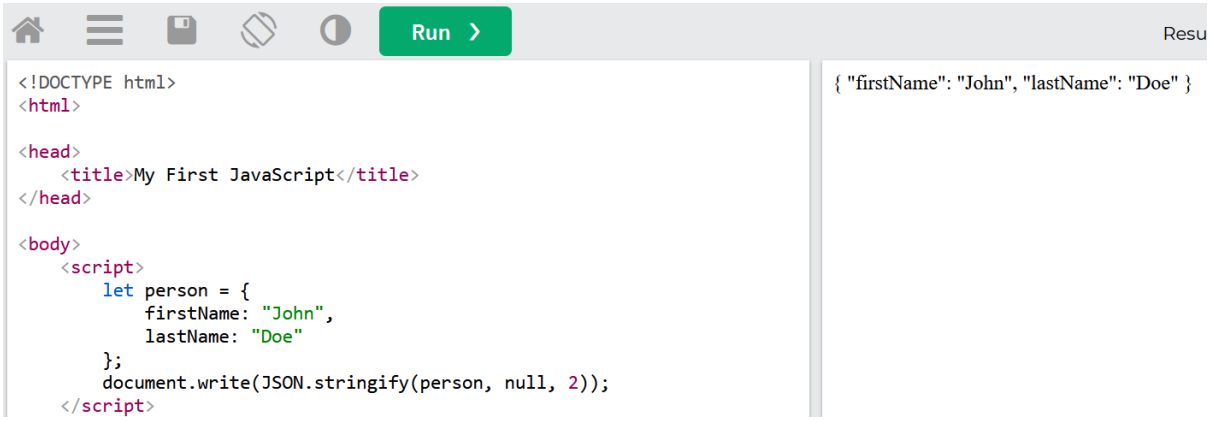
Example:

15. **JS Object Methods**: Object methods are functions defined inside objects, like `obj.sayHello()`.

Example:

16. **JS Object Display**: Objects can be displayed using `console.log()` or by serializing them with `JSON.stringify()`.

Example:

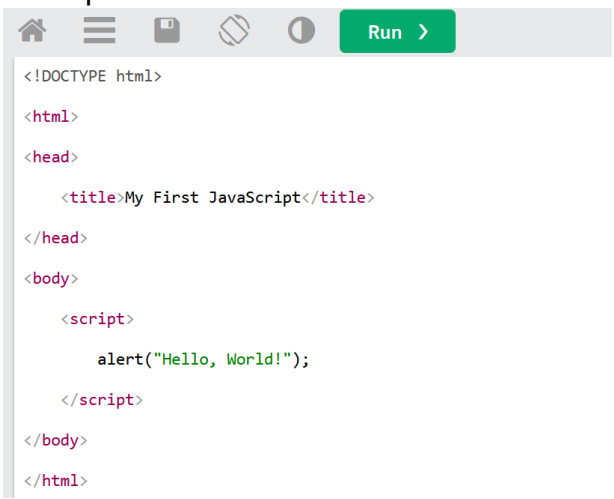


17. **JS Object Constructors**: Object constructors create new instances of an object, like `new Object()`.

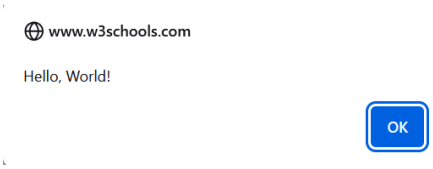
Example:

18. **JS Events**: Events are actions like clicks or keypresses, often handled with `addEventListener()`.

Example:

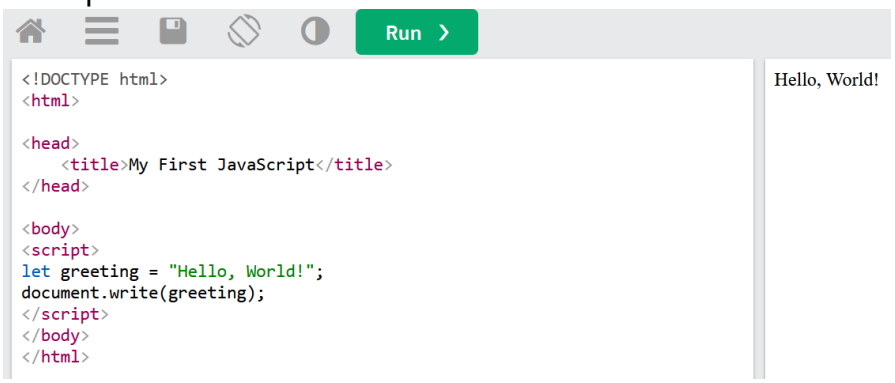


Output: A pop-up alert saying "Hello, World!":



19. JS Strings: Strings are sequences of characters, enclosed in single, double, or backticks.

Example:



20. JS String Methods: String methods include `.toUpperCase()`, `.substring()`, and `.indexOf()`.

Example:

21. JS String Templates: Template literals allow for string interpolation, e.g., `` `Hello ${name}` ``.

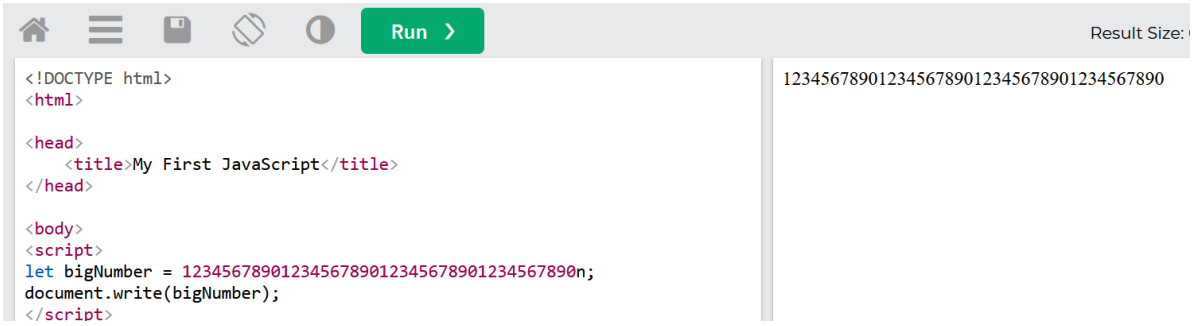
Example:

22. JS Numbers: Numbers are numeric values, such as `10` or `3.14`..

Example:

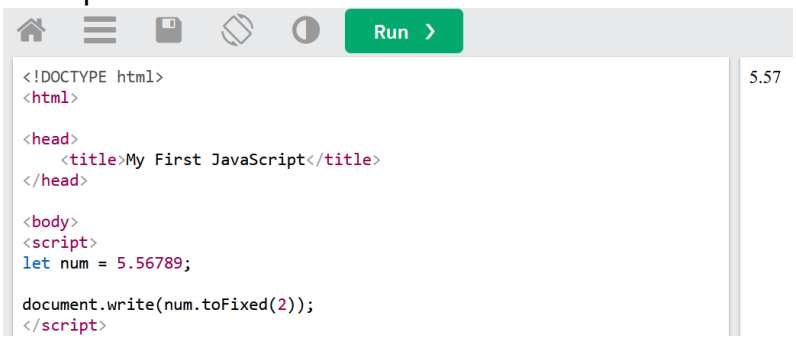
23. JS BigInt: BigInt allows for handling large integers, e.g., 12345678901234567890n.

Example:



24. JS Number Methods: Number methods include `.toFixed()` for decimal precision or `.toString()` to convert numbers to strings.

Example:

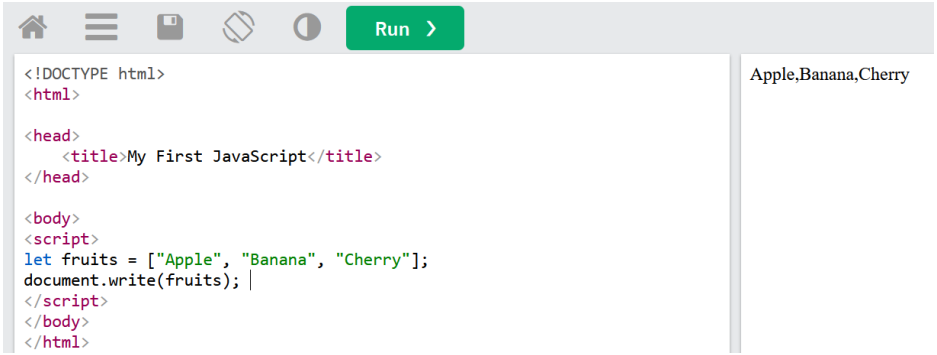


25. **JS Number Properties**: `Number.MAX_VALUE` represents the largest possible number in JavaScript.

Example:

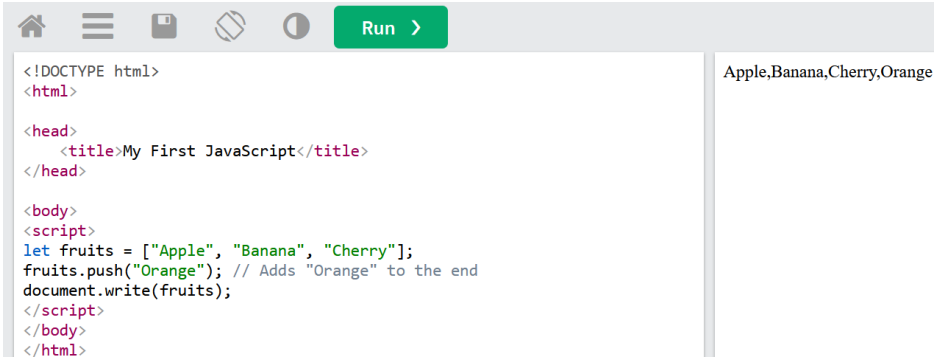
26. **JS Arrays**: Arrays are ordered collections of values, like `[1, 2, 3]`.

Example:



27. **JS Array Methods**: Array methods include `.push()`, `.pop()`, and `.shift()`.

Example:

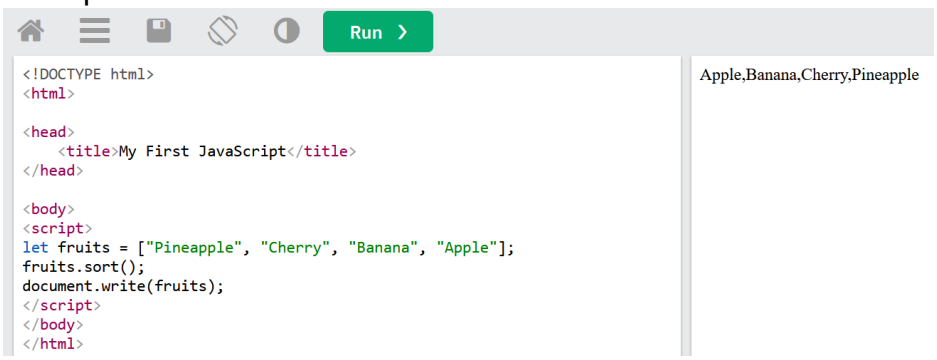


28. JS Array Search: `.indexOf()` and `.includes()` help find elements in arrays.

Example:

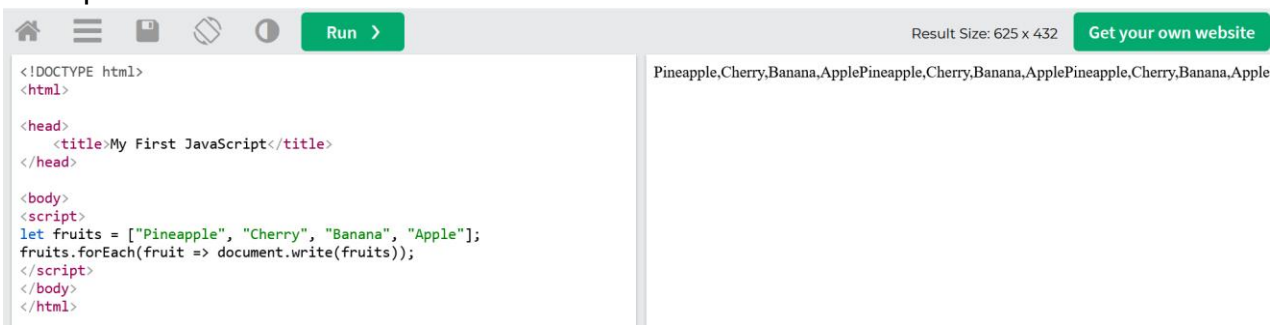
29. JS Array Sort: `.sort()` arranges array elements in order.

Example:



30. JS Array Iteration: `.forEach()` is used to iterate through array elements.

Example:

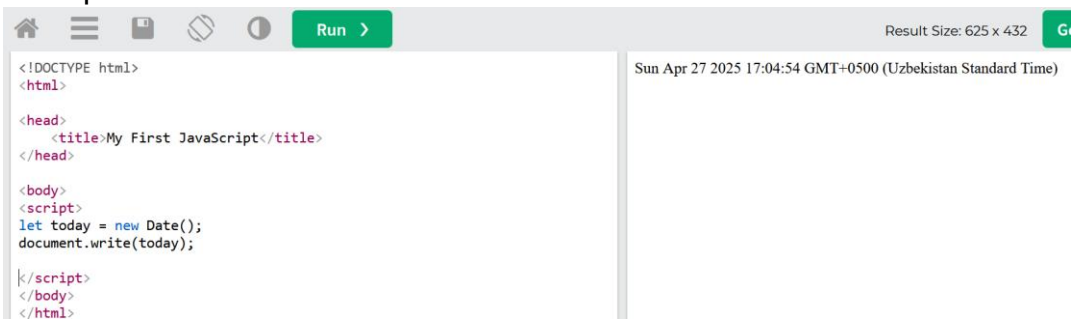


31. JS Array Const: Arrays can be declared as constants with `const`, but their elements can still be modified.

Example:

32. JS Dates: Dates are created with `new Date()`.

Example:



33. JS Date Formats: Date formats vary, but common ones include `Date.toLocaleDateString()` or `Date.toISOString()`.

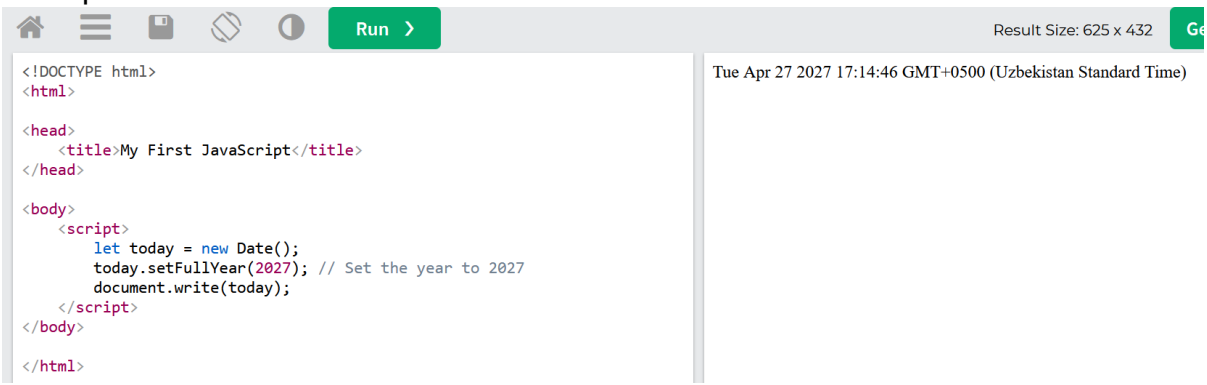
Example:

34. JS Date Get Methods: Methods like `.getFullYear()`, `.getMonth()`, and `.getDate()` retrieve date parts.

Example:

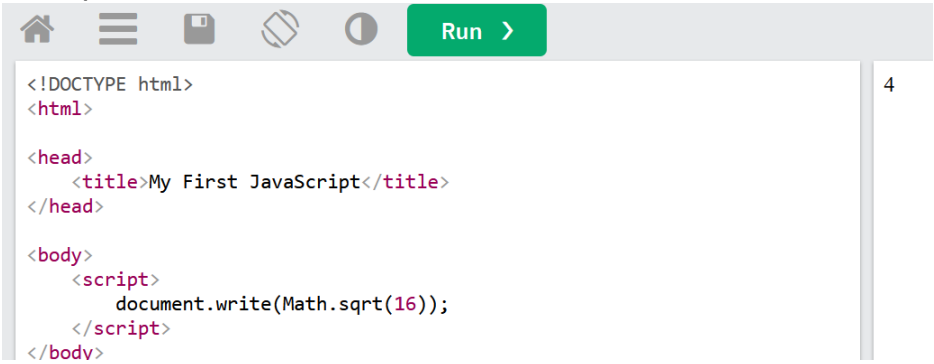
35. JS Date Set Methods: Methods like `.setFullYear()`, `.setMonth()`, and `.setDate()` set date parts.

Example:



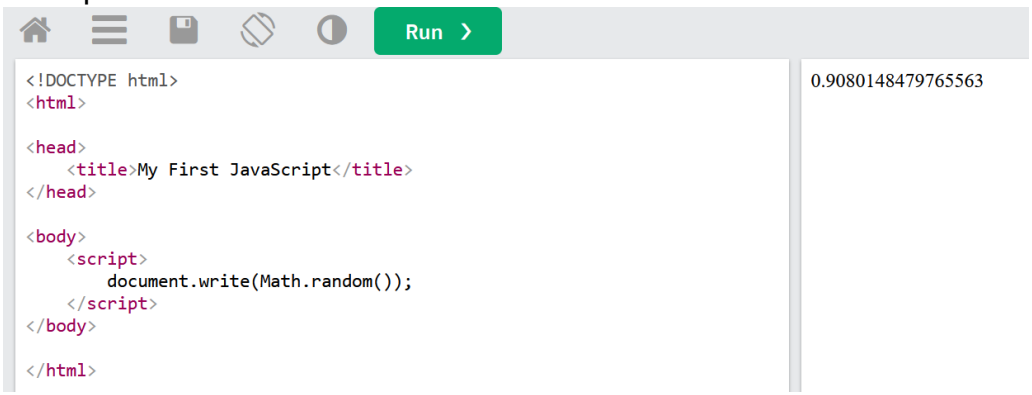
36. JS Math: The `Math` object provides mathematical functions like `Math.random()` or `Math.max()`.

Example:



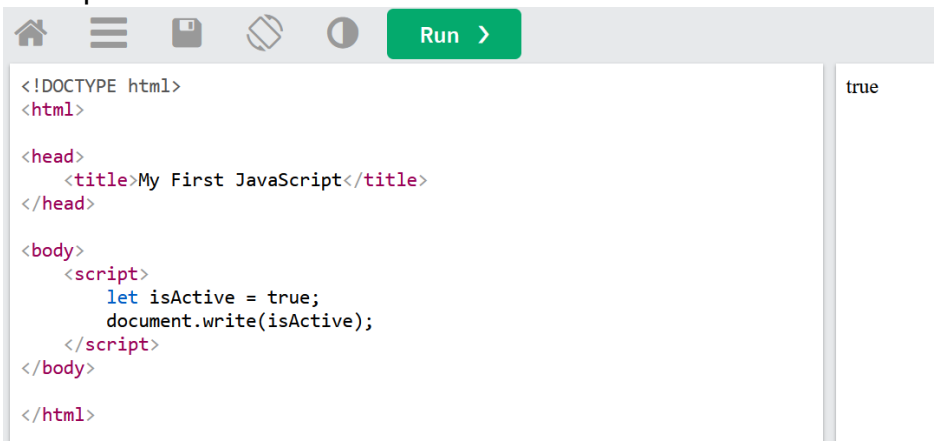
37. **JS Random**: `Math.random()` returns a random floating-point number between 0 and 1.

Example:



38. **JS Booleans**: Boolean values represent true or false, often used in conditionals.

Example:



39. JS Comparisons: Comparison operators like `==`, `===`, `!=`, and `!==` compare values.

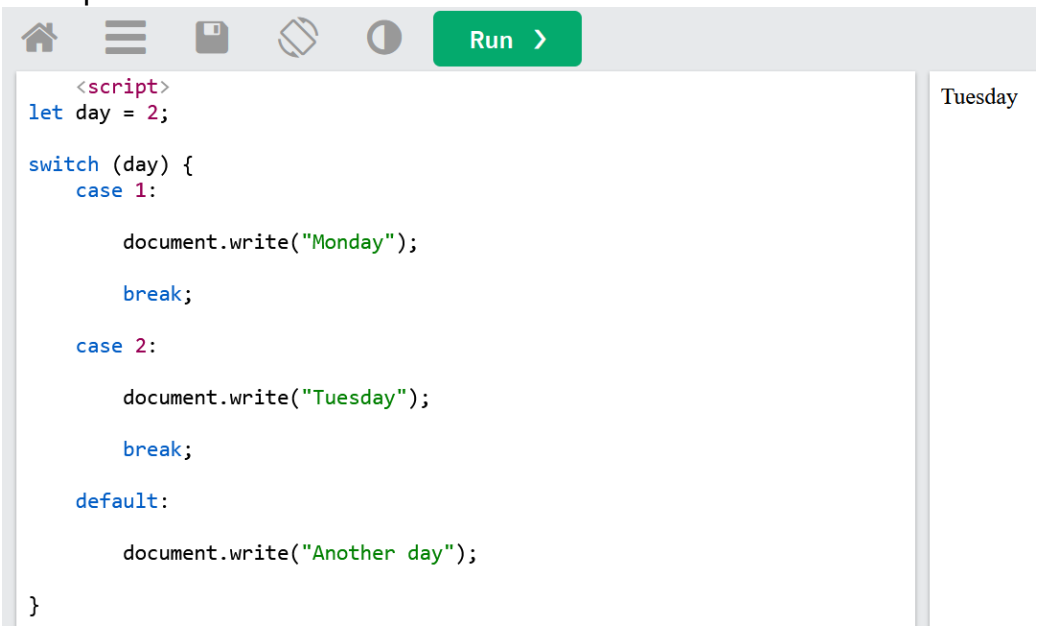
Example:

40. JS If Else: The `if-else` statement executes code based on a condition.

Example:

41. JS Switch: `switch` is used for multiple conditions, similar to a series of `if-else` statements.

Example:



42. JS Loop For: `for` loops iterate over a range of values.

Example:

43. JS Loop For In: The `for-in` loop iterates over the keys of an object.

Example:

44. JS Loop For Of: The `for-of` loop iterates over array values.

Example:

45. JS Loop While: The `while` loop runs as long as a condition is true.

Example:

46. JS Break: `break` is used to exit a loop or switch statement early.

Example:

47. JS Iterables: Iterables are objects that can be iterated over, such as arrays or sets.

Example:

48. JS Sets: A `Set` is a collection of unique values.

Example:

49. JS Set Methods: Methods for sets include `.add()`, `.delete()`, and `.has()`.

Example:

50. JS Maps: A `Map` is a collection of key-value pairs.

Example:

51. JS Map Methods: Map methods include `.set()`, `.get()`, and `.has()`.

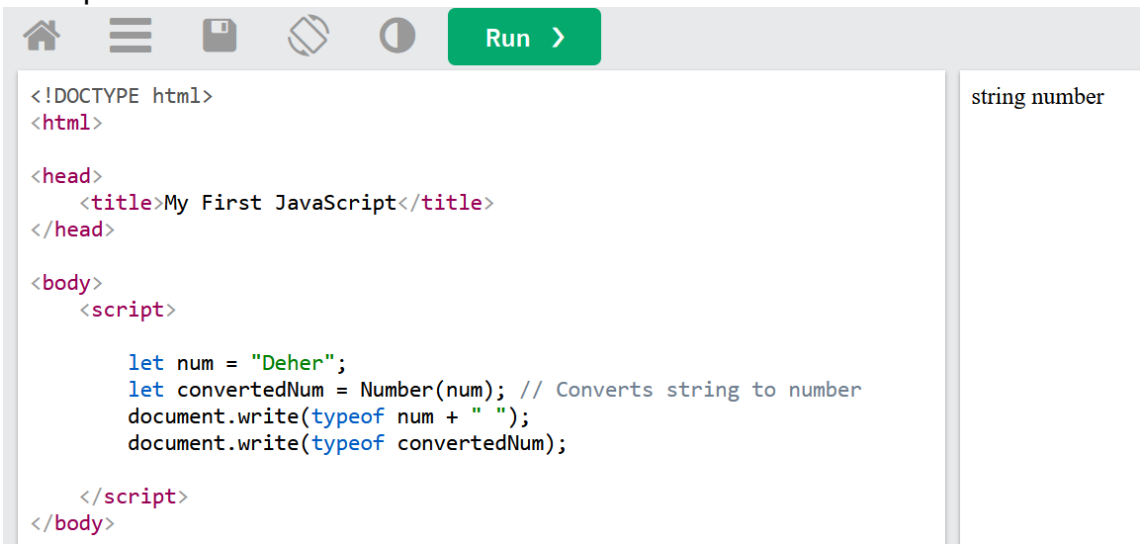
Example:

52. JS Typeof: `typeof` returns the type of a variable, like `typeof 10` returns `"number"`.

Example:

53. **JS Type Conversion**: Type conversion changes a value from one type to another, like `String(123)`.

Example:



54. **JS Destructuring**: Destructuring allows for unpacking values from arrays or objects.

Example:

55. JS Bitwise: Bitwise operators perform operations on binary representations of integers.

Example:

Explanation:

Bit Position	5 (0101)	3 (0011)	5 & 3 (Result)
1	1	1	1
2	0	1	0
3	1	0	0
4	0	0	0

Result in binary: `0001` → which is **1** in decimal.

56. JS RegExp: Regular expressions are used for pattern matching.

Example:

57. JS Precedence: Operator precedence determines the order of operations, such as * before +.

Example:

58. JS Errors: Errors can be handled using try, catch, and throw.

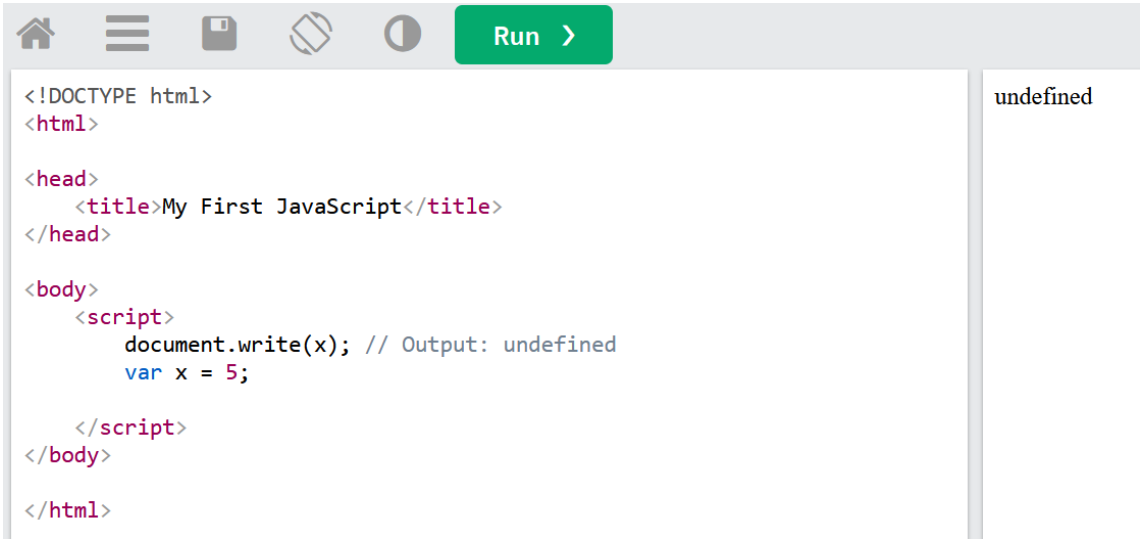
Example:

59. **JS Scope**: Scope refers to the context in which a variable is accessible, like global or local.

Example:

60. **JS Hoisting**: Hoisting is JavaScript's behavior of moving declarations to the top during execution.

Example:



61. **JS Strict Mode**: `"use strict";` enforces stricter parsing and error handling in JavaScript. In strict mode, if you try to assign a value to a variable without declaring it with let, const, or var, it will throw an error.

Example:

62. **JS this Keyword**: `this` refers to the current execution context, often the object that called the function.

Example:

63. JS Arrow Function: Arrow functions provide a shorter syntax for writing functions.

Example:

64. JS Classes: Classes in JavaScript are templates for creating objects.

Example:

65. **JS Modules**: Modules allow for splitting JavaScript code into reusable pieces, using `import` and `export`.

Example:

```
// In module.js
export const pi = 3.14;

// In main.js
import { pi } from './module.js';
console.log(pi); // Output: 3.14
```

66. **JS JSON**: JSON (JavaScript Object Notation) is a text format for representing objects, used for data exchange.

Example:

67. **JS Style Guide**: A style guide provides rules for writing consistent and readable code.

Example: Follow consistent naming conventions and indentation.

68. **JS Best Practices**: Best practices include writing clean, efficient, and readable code, using appropriate naming conventions, and avoiding global variables.

Example: Use `const` and `let` instead of `var` for variable declarations.

69. **JS Mistakes**: Common mistakes include using undeclared variables, improper scoping, or misunderstanding `this`.

Example: Forgetting to declare variables can lead to unexpected behavior.

70. **JS Performance**: Performance optimization includes techniques like minimizing DOM manipulations, using event delegation, and reducing file sizes.

Example: Minimize DOM manipulations and use event delegation.

71. **JS Reserved Words**: Reserved words like `break`, `class`, and `return` cannot be used as identifiers.

Example: `let` , `const` , `function` , `class` , etc. are reserved keywords.