



**Name: Deher Zainab**

**Sap ID: 49710**

**BSCS-5<sup>th</sup> 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:



```
<!DOCTYPE html>

<html>

<head>

  <title>My First JavaScript</title>

</head>

<body>

  <script>

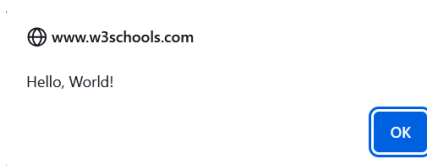
    alert("Hello, World!");

  </script>

</body>

</html>
```

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



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



```
<!DOCTYPE html>

<html>

<head>

  <title>My First JavaScript</title>

</head>

<body>

  <script>

    document.write("Hello, World!");

  </script>

</body>

</html>
```

Hello, World!

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

Example:



```
<html>
<head>
  <title>My First JavaScript</title>
</head>
<body>
  <script>
let x = 5;
let y = 10;
let z = x + y;

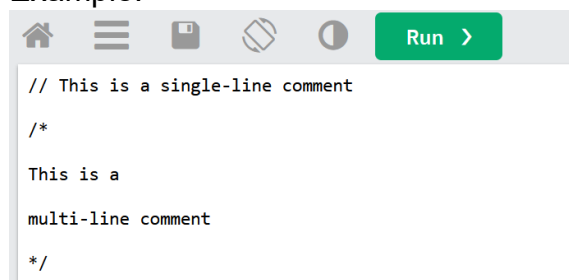
    document.write(z);
  </script>
```

15

#### 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:



```
// This is a single-line comment

/*
This is a
multi-line comment
*/
```

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

Example:



```
<!DOCTYPE html>
<html>
<head>
  <title>My First JavaScript</title>
</head>
<body>
  <script>
var name = "Alice";

    document.write(name);
  </script>
```

Alice

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

Example:



The screenshot shows a web browser interface. The address bar contains the text "10". The main content area displays the output of a JavaScript script. The script is as follows:

```
<!DOCTYPE html>
<html>
<head>
  <title>My First JavaScript</title>
</head>
<body>
  <script>
let x = 10;
if (true) {
  let x = 20; // different x
}

  document.write(x);
</script>
```

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

Example:



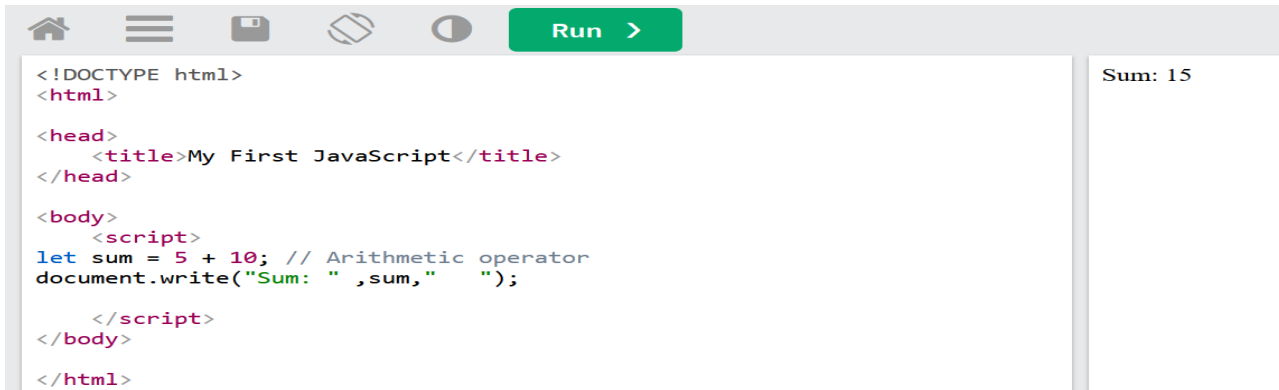
The screenshot shows a web browser interface. The address bar contains the text "10". The main content area displays the output of a JavaScript script. The script is as follows:

```
<!DOCTYPE html>
<html>
<head>
  <title>My First JavaScript</title>
</head>
<body>
  <script>
const PI = 3.14;
  </script>
</body>
</html>
```

## 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:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a green 'Run' button. The browser window displays the output of a JavaScript script: 'Sum: 15'. The code in the background is as follows:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
let sum = 5 + 10; // Arithmetic operator
document.write("Sum: ",sum," ");

  </script>
</body>
</html>
```

## Example 2:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a green 'Run' button. The browser window displays the output of a JavaScript script: '7'. The code in the background is as follows:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

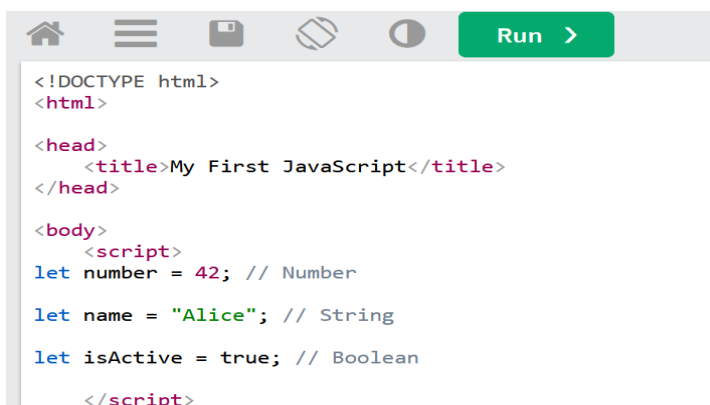
<body>
  <script>
let x = 5;

x += 2; // x = x + 2
document.write(x);

  </script>
</body>
```

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

## Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a green 'Run' button. The browser window displays the output of a JavaScript script. The code in the background is as follows:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
let number = 42; // Number

let name = "Alice"; // String

let isActive = true; // Boolean

  </script>
```

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, refresh, and a 'Run' button. The main content area displays HTML code with a JavaScript function `greet()` that returns the string `"Hello!"`. The function is called using `document.write(greet());`. The output on the right side of the browser is `Hello!`.

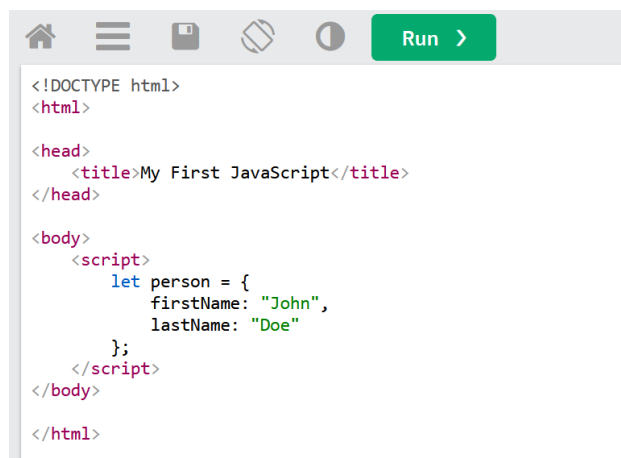
```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    function greet() {
      return "Hello!";
    }
    document.write(greet());
  </script>
</body>
</html>
```

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, refresh, and a 'Run' button. The main content area displays HTML code with a JavaScript object `person` defined using `let` and `const` syntax. The object has two properties: `firstName` with the value `"John"` and `lastName` with the value `"Doe"`. The output on the right side of the browser is empty.

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let person = {
      firstName: "John",
      lastName: "Doe"
    };
  </script>
</body>
</html>
```

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, refresh, and a 'Run' button. The main content area displays HTML code with a JavaScript object `person` defined using `let` and `const` syntax. The object has two properties: `firstName` with the value `"John"` and `lastName` with the value `"Doe"`. The object property `firstName` is accessed using dot notation (`person.firstName`) and the result is written to the document using `document.write(person.firstName);`. The output on the right side of the browser is `John`.

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let person = {
      firstName: "John",
      lastName: "Doe"
    };
    document.write(person.firstName);
  </script>
</body>
</html>
```

## 15. JS Object Methods: Object methods are functions defined inside objects, like

`obj.sayHello()`.

Example:



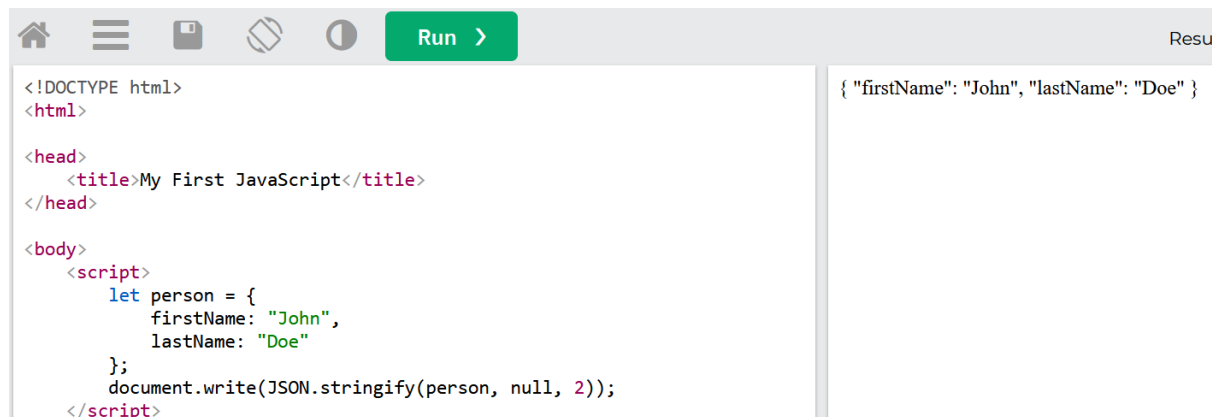
The screenshot shows a web browser interface. The address bar contains the text "John Doe". The main content area displays the following HTML and JavaScript code:

```
<title>My First JavaScript</title>
</head>
<body>
  <script>
    let person = {
      firstName: "John",
      lastName: "Doe",
      fullName: function() {
        return this.firstName + " " + this.lastName;
      }
    };

    document.write(person.fullName());
  </script>
```

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

Example:

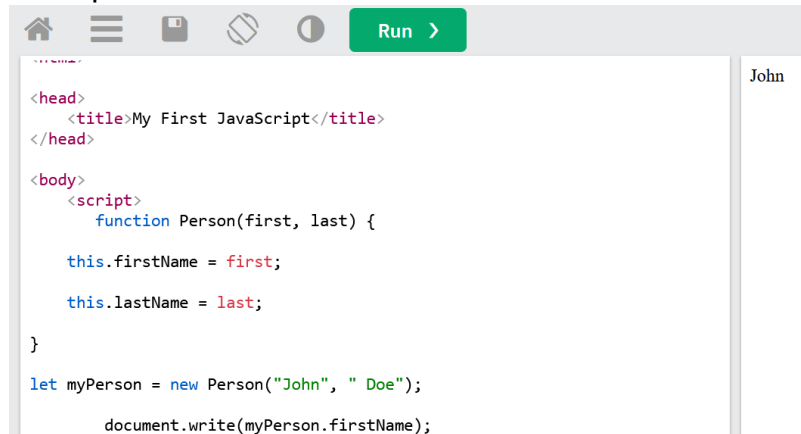


The screenshot shows a web browser interface. The address bar contains the text "Resu". The main content area displays the following HTML and JavaScript code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>My First JavaScript</title>
  </head>
  <body>
    <script>
      let person = {
        firstName: "John",
        lastName: "Doe"
      };
      document.write(JSON.stringify(person, null, 2));
    </script>
```

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

Example:



```
<!-- ... -->
<head>
  <title>My First JavaScript</title>
</head>

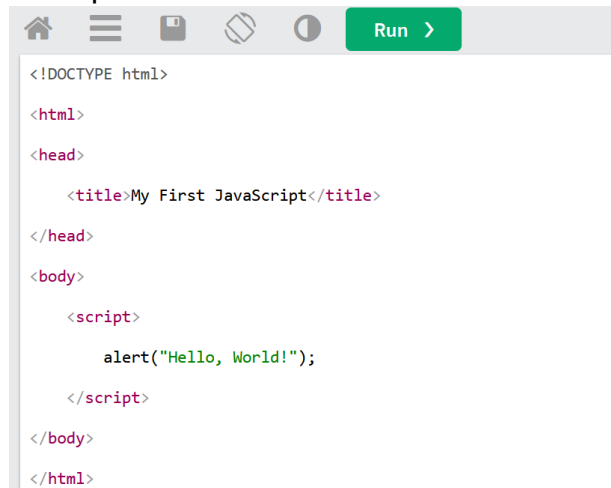
<body>
  <script>
    function Person(first, last) {
      this.firstName = first;
      this.lastName = last;
    }

    let myPerson = new Person("John", "Doe");

    document.write(myPerson.firstName);
  </script>
</body>
</html>
```

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

Example:



```
<!DOCTYPE html>
<html>
<head>
  <title>My First JavaScript</title>
</head>
<body>
  <script>
    alert("Hello, World!");
  </script>
</body>
</html>
```

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





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

Example:



```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
<script>
let greeting = "Hello, World!";
document.write(greeting);
</script>
</body>
</html>
```

Hello, World!

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

Example:



```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

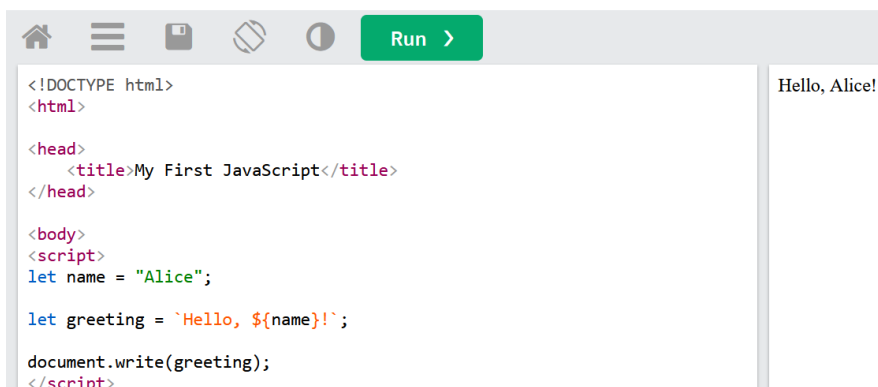
<body>
<script>
let text = "Hello";

document.write(text.toUpperCase());
</script>
</body>
</html>
```

HELLO

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

Example:



```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
<script>
let name = "Alice";

let greeting = `Hello, ${name}!`;

document.write(greeting);
</script>
```

Hello, Alice!

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a 'Run' button. The main content area displays the output of a JavaScript script: the number 25. The code in the background is as follows:

```
<!DOCTYPE html>
<html>

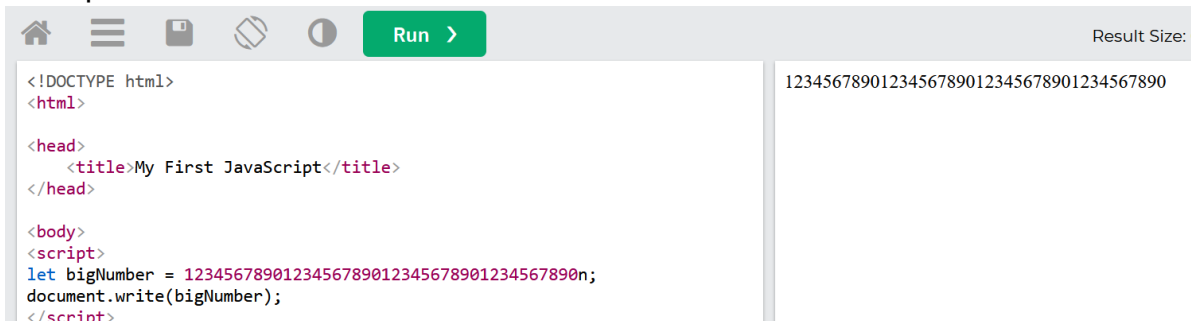
<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let age = 25; // Number

    document.write(age);
  </script>
</body>
</html>
```

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a 'Run' button. The main content area displays the output of a JavaScript script: the large integer 1234567890123456789012345678901234567890. The code in the background is as follows:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let bigNumber = 1234567890123456789012345678901234567890n;
    document.write(bigNumber);
  </script>
```

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a 'Run' button. The main content area displays the output of a JavaScript script: the number 5.57. The code in the background is as follows:

```
<!DOCTYPE html>
<html>

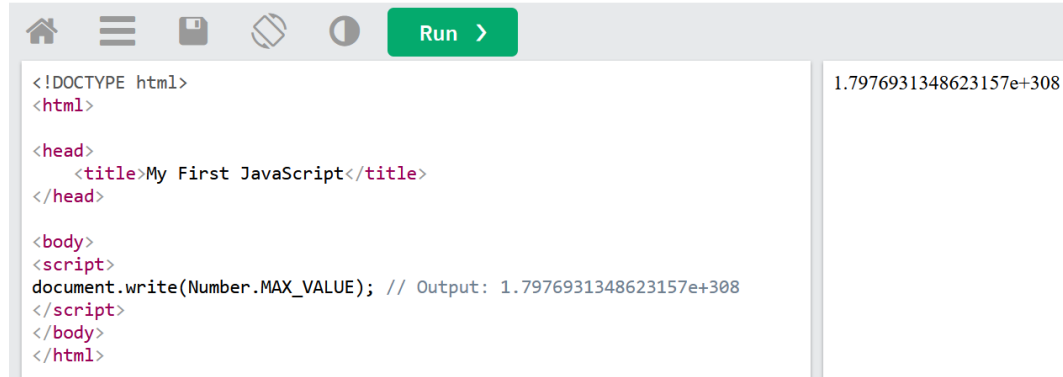
<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let num = 5.56789;

    document.write(num.toFixed(2));
  </script>
```

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a green 'Run' button. The browser window displays the output of a JavaScript script. The code in the background is as follows:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
<script>
document.write(Number.MAX_VALUE); // Output: 1.7976931348623157e+308
</script>
</body>
</html>
```

The output displayed in the browser is: 1.7976931348623157e+308

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a green 'Run' button. The browser window displays the output of a JavaScript script. The code in the background is as follows:

```
<!DOCTYPE html>
<html>

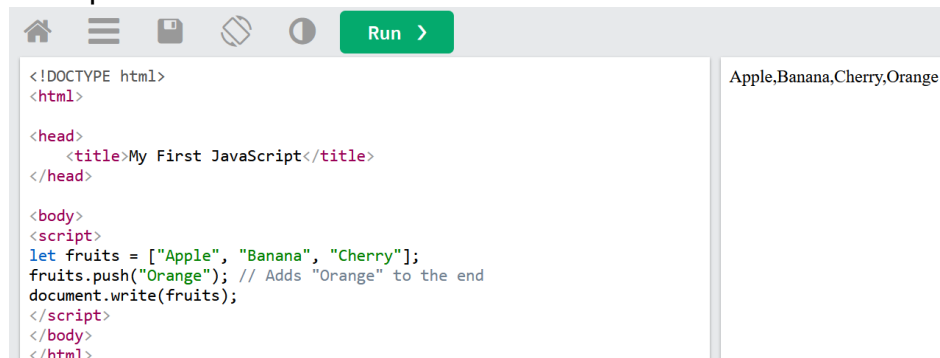
<head>
  <title>My First JavaScript</title>
</head>

<body>
<script>
let fruits = ["Apple", "Banana", "Cherry"];
document.write(fruits);
</script>
</body>
</html>
```

The output displayed in the browser is: Apple,Banana,Cherry

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a green 'Run' button. The browser window displays the output of a JavaScript script. The code in the background is as follows:

```
<!DOCTYPE html>
<html>

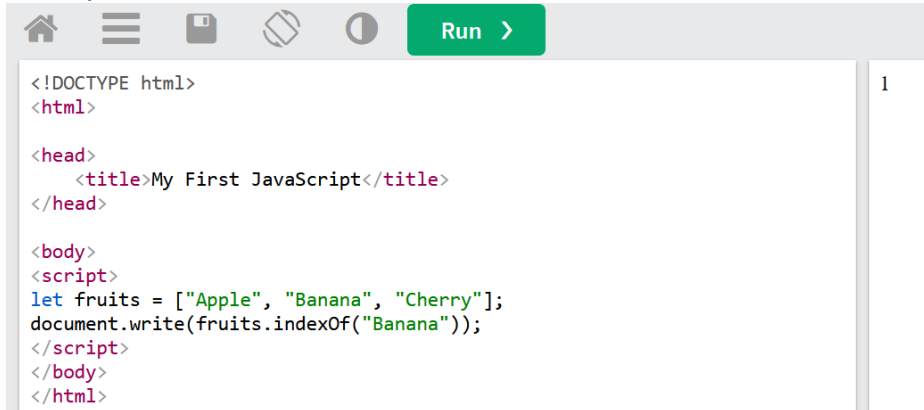
<head>
  <title>My First JavaScript</title>
</head>

<body>
<script>
let fruits = ["Apple", "Banana", "Cherry"];
fruits.push("Orange"); // Adds "Orange" to the end
document.write(fruits);
</script>
</body>
</html>
```

The output displayed in the browser is: Apple,Banana,Cherry,Orange

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a 'Run' button. The browser window displays the output of a JavaScript script. The script defines an array of fruits and uses `.indexOf()` to find the index of 'Banana'. The output '1' is displayed in the browser window.

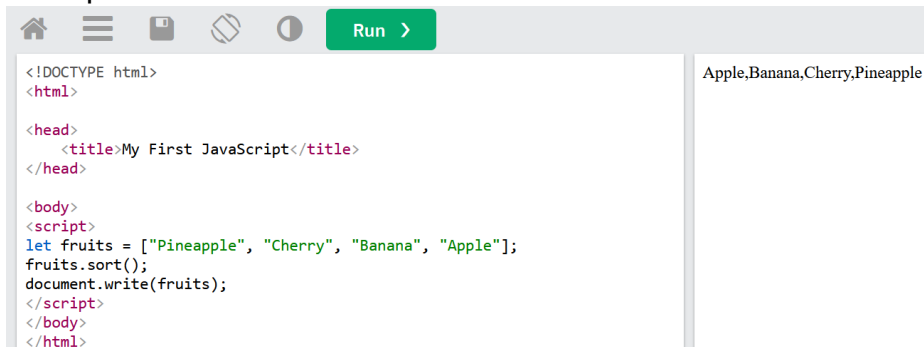
```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
let fruits = ["Apple", "Banana", "Cherry"];
document.write(fruits.indexOf("Banana"));
  </script>
</body>
</html>
```

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a 'Run' button. The browser window displays the output of a JavaScript script. The script defines an array of fruits and uses `.sort()` to sort the array. The output 'Apple,Banana,Cherry,Pineapple' is displayed in the browser window.

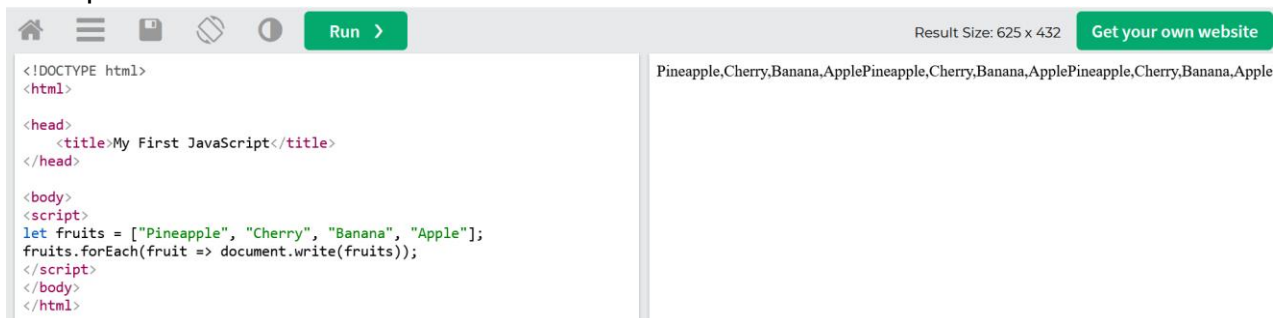
```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
let fruits = ["Pineapple", "Cherry", "Banana", "Apple"];
fruits.sort();
document.write(fruits);
  </script>
</body>
</html>
```

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a 'Run' button. The browser window displays the output of a JavaScript script. The script defines an array of fruits and uses `.forEach()` to iterate through the array and write each element. The output 'Pineapple,Cherry,Banana,ApplePineapple,Cherry,Banana,ApplePineapple,Cherry,Banana,Apple' is displayed in the browser window. The toolbar also shows 'Result Size: 625 x 432' and a 'Get your own website' button.

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
let fruits = ["Pineapple", "Cherry", "Banana", "Apple"];
fruits.forEach(fruit => document.write(fruits));
  </script>
</body>
</html>
```

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a 'Run' button. The browser window displays an HTML document with the following code:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

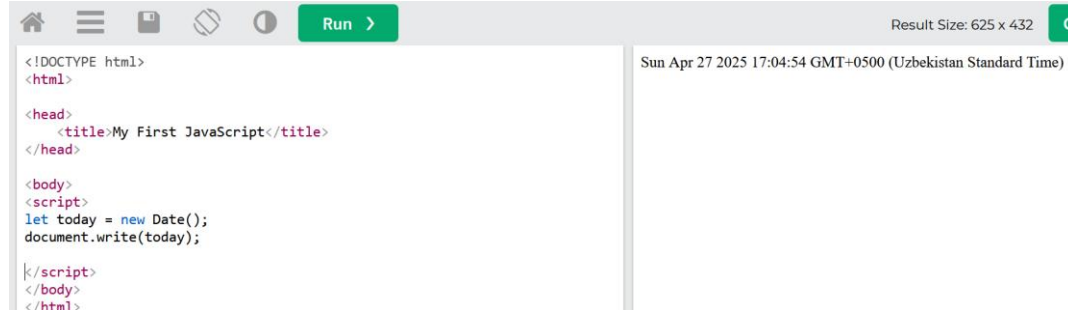
<body>
  <script>
    const colors = ["Red", "Green", "Blue"];
    document.write(colors);

    // |colors = ["Yellow"]; // This will throw an error
  </script>
</body>
</html>
```

The output of the script is displayed on the right side of the browser window: "Red,Green,Blue".

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a 'Run' button. The browser window displays an HTML document with the following code:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let today = new Date();
    document.write(today);
  </script>
</body>
</html>
```

The output of the script is displayed on the right side of the browser window: "Sun Apr 27 2025 17:04:54 GMT+0500 (Uzbekistan Standard Time)".

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a 'Run' button. The browser window displays an HTML document with the following code:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let today = new Date();
    document.write(today.toDateString());
  </script>
</body>
</html>
```

The output of the script is displayed on the right side of the browser window: "Sun Apr 27 2025".

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

Example:



The screenshot shows a web browser window with a light gray header bar containing icons for home, menu, save, print, and a 'Run' button. The main content area displays the output of a JavaScript script. The script is as follows:

```
<!DOCTYPE html>
<html>

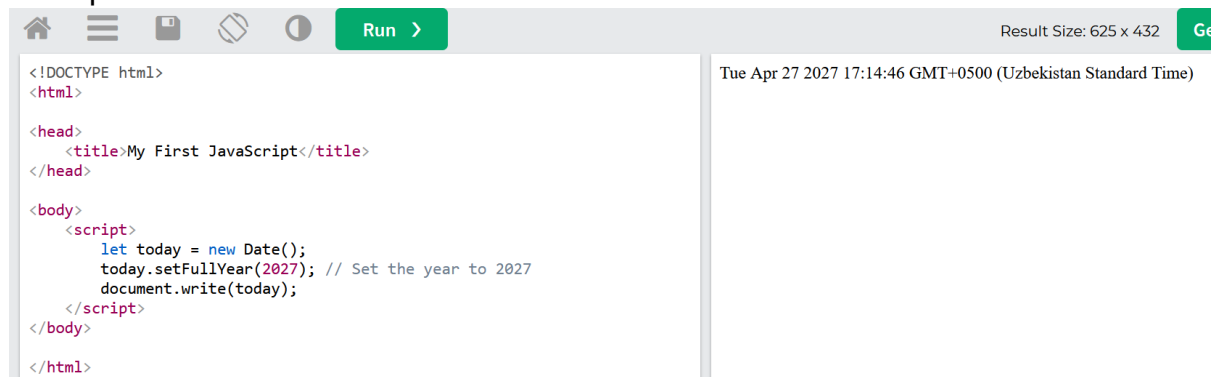
<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let today = new Date();
    document.write(today.getFullYear());
  </script>
</body>
</html>
```

The output of the script is the year '2025'.

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

Example:



The screenshot shows a web browser window with a light gray header bar containing icons for home, menu, save, print, and a 'Run' button. The main content area displays the output of a JavaScript script. The script is as follows:

```
<!DOCTYPE html>
<html>

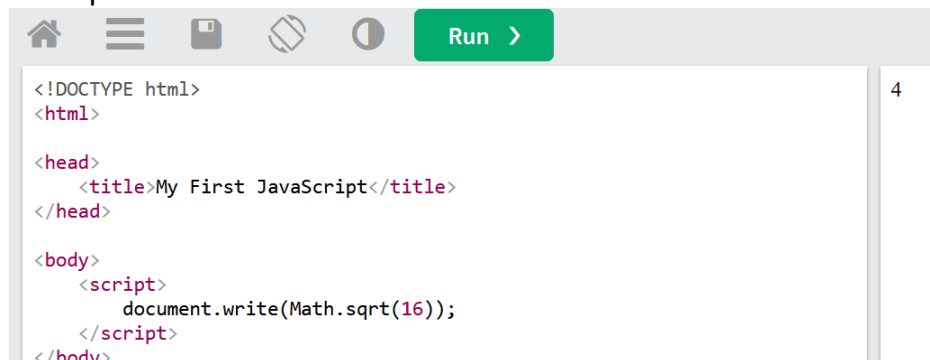
<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let today = new Date();
    today.setFullYear(2027); // Set the year to 2027
    document.write(today);
  </script>
</body>
</html>
```

The output of the script is 'Tue Apr 27 2027 17:14:46 GMT+0500 (Uzbekistan Standard Time)'. The browser's status bar at the bottom right shows 'Result Size: 625 x 432'.

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

Example:



The screenshot shows a web browser window with a light gray header bar containing icons for home, menu, save, print, and a 'Run' button. The main content area displays the output of a JavaScript script. The script is as follows:

```
<!DOCTYPE html>
<html>

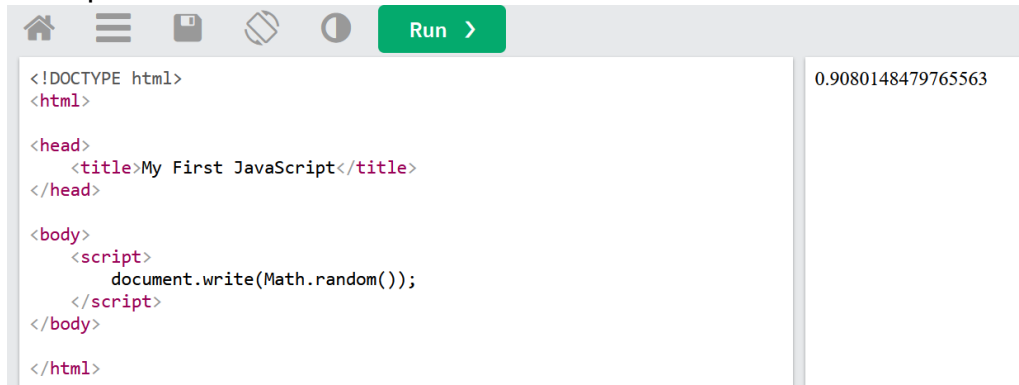
<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    document.write(Math.sqrt(16));
  </script>
</body>
```

The output of the script is the number '4'.

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a green 'Run >' button. The main content area displays the following HTML code:

```
<!DOCTYPE html>
<html>

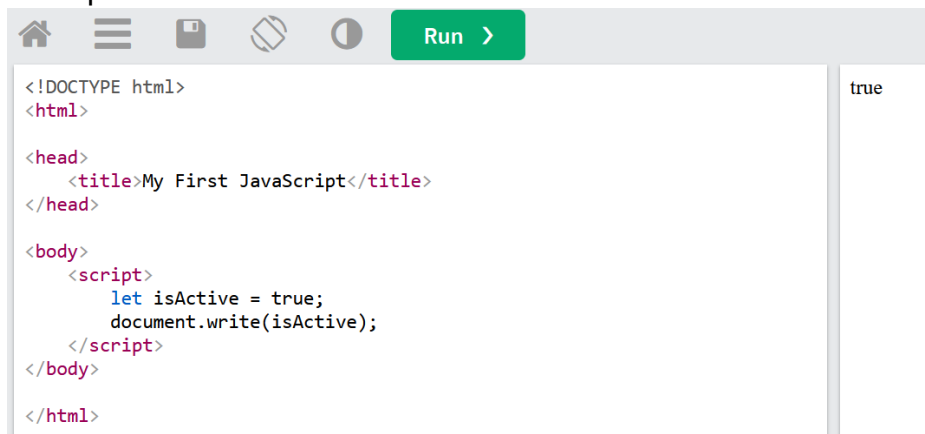
<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    document.write(Math.random());
  </script>
</body>
</html>
```

To the right of the code, the output of the script is displayed as the decimal value 0.9080148479765563.

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a green 'Run >' button. The main content area displays the following HTML code:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let isActive = true;
    document.write(isActive);
  </script>
</body>
</html>
```

To the right of the code, the output of the script is displayed as the boolean value true.

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

Example:



```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let isEqual = (5 === 5);
    let isNotEqual = (5 === 8);
    document.write(isEqual);
    document.write(" ");
    document.write(isNotEqual);

  </script>
</body>
```

true false

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

Example:



```
<!-- ... -->
<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let age = 18;
    if (age >= 18) {
      document.write("Adult");
    } else {
      document.write("Minor");
    }

  </script>
```

Adult



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

Example:



The screenshot shows a code editor with a toolbar at the top containing icons for home, menu, save, undo, redo, and a green 'Run' button. The code in the editor is as follows:

```
<script>
let day = 2;

switch (day) {
  case 1:

    document.write("Monday");

    break;

  case 2:

    document.write("Tuesday");

    break;

  default:

    document.write("Another day");
}
```

The output of the code, displayed on the right side of the editor, is "Tuesday".

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

Example:



The screenshot shows a code editor with a toolbar at the top containing icons for home, menu, save, undo, redo, and a green 'Run' button. The code in the editor is as follows:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    for (let i = 0; i < 5; i++) {

      document.write(i);

    }

  </script>
</body>
</html>
```

The output of the code, displayed on the right side of the editor, is "01234".

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

Example:



The screenshot shows a web browser interface with a code editor on the left and a rendered page on the right. The code editor contains HTML and JavaScript code. The JavaScript code uses a `for-in` loop to iterate over the keys of an object `person` and write them to the document. The rendered page shows the output: "name: Alice age: 25".

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

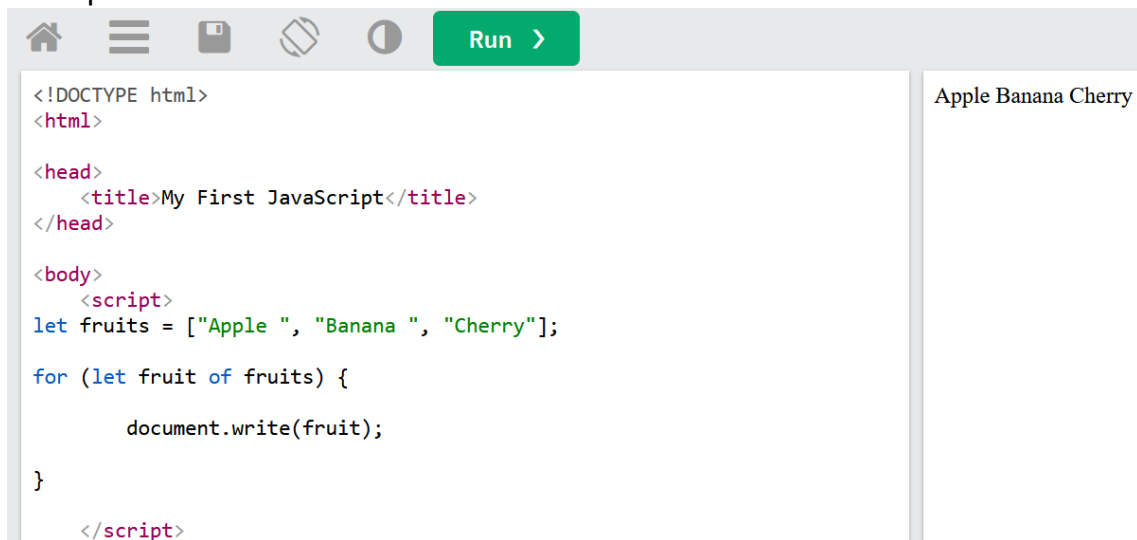
<body>
  <script>
let person = {name: "Alice ", age: 25};
for (let key in person) {
    document.write(key + ": " + person[key]);
}

  </script>
</body>
</html>
```

name: Alice age: 25

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

Example:



The screenshot shows a web browser interface with a code editor on the left and a rendered page on the right. The code editor contains HTML and JavaScript code. The JavaScript code uses a `for-of` loop to iterate over the values of an array `fruits` and write them to the document. The rendered page shows the output: "Apple Banana Cherry".

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
let fruits = ["Apple ", "Banana ", "Cherry"];
for (let fruit of fruits) {
    document.write(fruit);
}

  </script>
</body>
</html>
```

Apple Banana Cherry

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a green 'Run' button. The browser window displays the output of a JavaScript script. The script is as follows:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
let count = 0;

while (count < 5) {

  document.write(count);

  count++;

}
```

The output displayed in the browser is the sequence of numbers 0, 1, 2, 3, and 4, which are concatenated together to form the string '01234'.

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a green 'Run' button. The browser window displays the output of a JavaScript script. The script is as follows:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
for (let i = 0; i < 10; i++) {

  if (i === 7) break;

  document.write(i);

}
```

The output displayed in the browser is the sequence of numbers 0, 1, 2, 3, 4, 5, and 6, which are concatenated together to form the string '0123456'. The loop is terminated early when `i` reaches 7.

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

Example:



The code editor shows a file named '123'. The code is as follows:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
let iterable = [1, 2, 3];

for (let value of iterable) {
  document.write(value);
}
```

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

Example:



The code editor shows a file named '1,2,3'. The code is as follows:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let mySet = new Set([1, 2, 3, 3]);
    document.write(Array.from(mySet)); // Output: 1,2,3
  </script>
</body>
</html>
```

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

Example:



The screenshot shows a web browser window with a toolbar at the top containing icons for home, menu, save, print, and a green 'Run' button. The browser's address bar is empty. The main content area displays the output of a JavaScript script: 'true'. The script, visible in the background, creates a Set with the values [1, 2, 3, 3], adds the value 4, and then checks if the value 2 is in the set, which returns true.

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let mySet = new Set([1, 2, 3, 3]);
    mySet.add(4);
    document.write(mySet.has(2));
  </script>
</body>

</html>
```

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

Example:



The screenshot shows a web browser window with a toolbar at the top containing icons for home, menu, save, print, and a green 'Run' button. The browser's address bar is empty. The main content area displays the output of a JavaScript script: 'Alice'. The script, visible in the background, creates a new Map, sets the value 'Alice' for the key 'name', and then retrieves the value for the key 'name', which is 'Alice'.

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let myMap = new Map();
    myMap.set("name", "Alice");
    document.write(myMap.get("name"));
  </script>
</body>

</html>
```

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

Example:



```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let myMap = new Map();
    myMap.set("name", "Alice");
    myMap.delete("name");
    document.write(myMap.has("name"));
  </script>
</body>

</html>
```

false

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

Example:



```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>


<body>
  <script>
    document.write(typeof "Hello");
  </script>
</body>

</html>
```

string

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

Example:



The screenshot shows a web editor interface with a toolbar at the top containing icons for home, menu, save, undo, redo, and a 'Run' button. The code area contains the following HTML and JavaScript:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>

    let num = "Deher";
    let convertedNum = Number(num); // Converts string to number
    document.write(typeof num + " ");
    document.write(typeof convertedNum);

  </script>
</body>
```

The output area on the right displays the result of the JavaScript execution: "string number".

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

Example:



The screenshot shows a web editor interface with a toolbar at the top containing icons for home, menu, save, undo, redo, and a 'Run' button. The code area contains the following HTML and JavaScript:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>

    let arr = [1, 2, 3];
    let [a, b] = arr; // Destructuring: a=1, b=2
    document.write("a = " + a + "<br>");
    document.write("b = " + b);

  </script>
</body>

</html>
```

The output area on the right displays the result of the JavaScript execution: "a = 1" followed by "b = 2" on a new line.

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

Example:



The screenshot shows a web browser window with a toolbar at the top containing icons for home, menu, save, print, and a green 'Run >' button. The browser's address bar shows the number '1'. The main content area displays the following HTML and JavaScript code:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let result = 5 & 3; // Bitwise AND
    document.write(result);
  </script>
</body>
</html>
```

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:



The screenshot shows a web browser window with a toolbar at the top containing icons for home, menu, save, print, and a green 'Run >' button. The browser's address bar shows the word 'true'. The main content area displays the following HTML and JavaScript code:

```
<!DOCTYPE html>
<html>

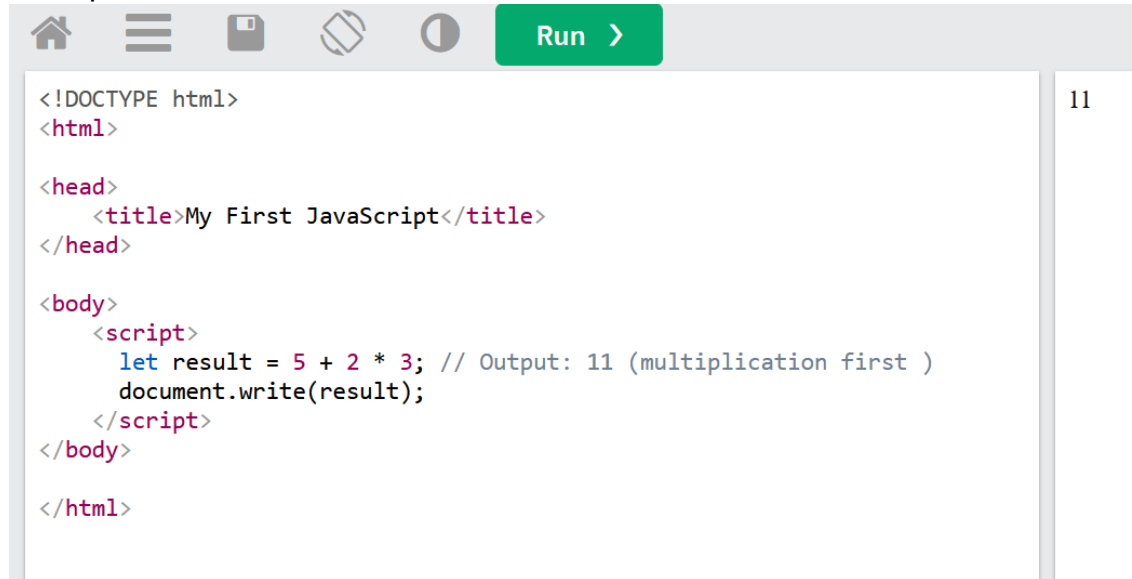
<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let regex = /abc/;
    document.write(regex.test("abcdef"));
  </script>
</body>
</html>
```



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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a 'Run' button. The main content area displays HTML code with a JavaScript script. The script calculates the result of 5 + 2 \* 3, which is 11, and writes it to the document. The output '11' is visible on the right side of the browser window.

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let result = 5 + 2 * 3; // Output: 11 (multiplication first )
    document.write(result);
  </script>
</body>

</html>
```

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

Example:



The screenshot shows a web browser interface with a toolbar at the top containing icons for home, menu, save, print, and a 'Run' button. The main content area displays HTML code with a JavaScript script. The script uses a try-catch block to throw an error with the message 'An error occurred' and then writes this message to the document. The output 'An error occurred' is visible on the right side of the browser window.

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    try {
      throw new Error("An error occurred");
    } catch (e) {
      document.write(e.message); // Output: An error occurred
    }
  </script>
</body>
```

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

Example:



The screenshot shows a code editor with a toolbar at the top containing icons for home, menu, save, undo, and a green 'Run' button. The code is written in HTML with embedded JavaScript. The JavaScript code defines a variable `x` with the value 10, creates a function `myFunction` that defines a local variable `y` with the value 5 and writes `y` to the document, and then calls `myFunction` after writing `x` to the document. The output on the right side of the editor is `x = 10, y = 5`.

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

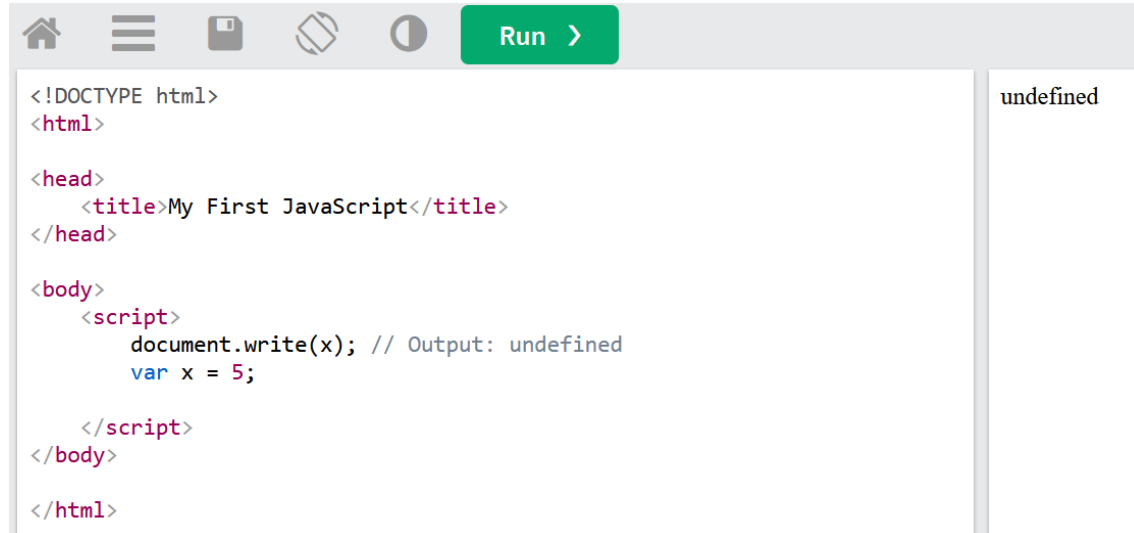
<body>
  <script>
    let x = 10;
    function myFunction() {
      let y = 5;
      document.write(" , y = " + y);
    }

    document.write("x = " + x);
    myFunction();
  </script>
</body>
```

x = 10, y = 5

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

Example:



The screenshot shows a code editor with a toolbar at the top containing icons for home, menu, save, undo, and a green 'Run' button. The code is written in HTML with embedded JavaScript. The JavaScript code attempts to write the value of `x` to the document before declaring `x`. The output on the right side of the editor is `undefined`, demonstrating the effect of hoisting.

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    document.write(x); // Output: undefined
    var x = 5;
  </script>
</body>
</html>
```

undefined

**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:



```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    "use strict";
    x = 3.14; // Throws an error because x is not declared
    document.write(x);

  </script>
</body>

</html>
```

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

Example:



```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    let obj = {
      name: "Alice",
      greet: function() {
        document.write("Hello, " + this.name);
      }
    };

    obj.greet();
  </script>
```

Hello, Alice

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

Example:



The screenshot shows a code editor interface with a toolbar at the top containing icons for home, menu, save, undo, and redo, along with a green 'Run' button. The code is written in HTML and JavaScript. It defines an arrow function 'add' and uses it to calculate the sum of 2 and 3, displaying the result '5' in the browser window.

```
<!DOCTYPE html>
<html>

<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
    const add = (a, b) => a + b;
    document.write(add(2, 3));
  </script>
</body>
```

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

Example:



The screenshot shows a code editor interface with a toolbar at the top containing icons for home, menu, save, undo, and redo, along with a green 'Run' button. The code is written in HTML and JavaScript. It defines a 'Person' class with a constructor and a 'greet' method. An instance of the class is created with the name 'Alice', and the 'greet' method is called, displaying 'Hello, Alice' in the browser window.

```
<head>
  <title>My First JavaScript</title>
</head>

<body>
  <script>
class Person {
  constructor(name) {
    this.name = name;
  }
  greet() {
    document.write("Hello, " + this.name);
  }
}

let person = new Person("Alice");
person.greet();

  </script>
```

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

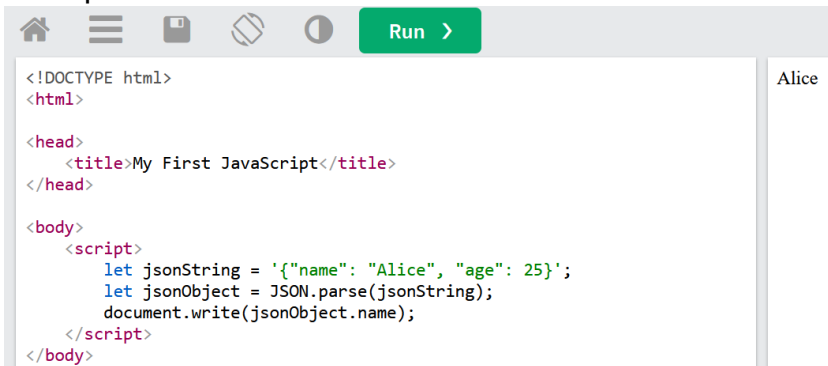
Example:

```
// 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.