Terminologies

1. The three ways to declare a variable are var, let, and const.
2. You should avoid using var because let and const have block scope while var has function scope.
3. When naming variables, you should use descriptive and meaningful names, avoid using reserved words, and stick to a naming convention (e.g., camelCase).
4. When using the + operator with numbers and strings, it is important to be aware of the data types. When both operands are numbers, the operator performs arithmetic addition. When one of the operands is a string, the operator performs string concatenation.
5. The % operator (modulo) returns the remainder of the division of the first operand by the second operand.
6. The double equals (==) operator performs type coercion, meaning it tries to convert the operands to the same type before comparison. The triple equals (===) operator does not perform type coercion and only returns true if both operands have the same type and value.
7. NaN (Not a Number) is returned when a mathematical operation is performed on a non-numeric value.
8. You can increment a number by adding 1 (e.g., x++) and decrement a number by subtracting 1 (e.g., x--).
9. The prefix increment operator (++x) increments the value of the variable and then returns the new value. The post-fix increment operator (x++) returns the original value of the variable and then increments the value.
10. Operator precedence determines the order in which operations are performed in an expression. JavaScript follows the order of operations defined in the specification. You can use parentheses to control the precedence of operations in an expression.
11. You can log information to the console using the console.log() method.
12. The unary plus operator (+) converts a string representation of an integer to a number.
13. The eight data types in JavaScript are: number, string, boolean, null, undefined, object, symbol, and big int.
14. The object data type is not primitive.
15. null represents a deliberate non-value, while undefined represents a lack of value.
16. Single quotes ('), double quotes ("), and backticks (`) can all be used to declare strings, but backticks allow for embedded expressions and are used for template literals.
17. The term for embedding variables/expressions in a string is string interpolation.
18. Backtick quotes (`) let you embed variables/expressions in a string.
19. You can embed variables/expressions in a string using template literals and placeholders (e.g. ${expression}).
20. You can escape characters in a string by adding a backslash (\) before the character.
21. The slice() method returns a portion of a string, specified by start and end indexes. The substring() method is similar to slice, but it can only accept positive indexes. The substr() method returns a portion of a string, specified by the start index and the number of characters.
22. The three logical operators are && (and), || (or), and ! (not).

23. Comparison operators are used to compare two values and return a boolean value of true or false. These operators include == (equal to), != (not equal to), > (greater than), < (less than), >= (greater than or equal to), and <= (less than or equal to).
24. Truthy and false values are values in JavaScript that are treated as either true or false.
25. The false values in JavaScript are false, 0, "", null, undefined, and NaN.
26. Conditionals are statements that control the flow of code execution based on whether a condition is true or false.
27. The syntax for an if/else conditional is:

```
if (condition) {
  // if condition = true
} else {
  // if condition = false
}
```

28. The syntax for a switch statement is:

```
switch (expression) {
  case value1:
    // code to be executed if the expression matches value1
    break;
  case value2:
    // code to be executed if the expression matches value2
    break;
}
```

29. The syntax for a ternary operator is:

```
condition ? valueIfTrue : valueIfFalse;
```

30. Nesting refers to the practice of placing one statement inside another. This is often used to create complex conditions or to execute multiple statements based on a single condition.
31. Functions are useful for organizing and reusing code. They can take input parameters, perform a set of operations, and return a value.
32. To invoke a function, simply call the function by its name, followed by parenthesis to specify any input parameters. For example:

```
function myFunction() {
  console.log("Hello World");
}

myFunction(); // outputs "Hello World"
```

33. Anonymous functions are functions that are defined without a name. These can be stored as variables or passed as arguments to other functions.

34. Function scope refers to the visibility and accessibility of variables and functions within a function. Variables declared inside a function are only accessible within that function and are not visible outside of it.
35. Return values are the values that a function returns when it is finished executing. A function can return any type of data, including primitive values, objects, and arrays.
36. Arrow functions are a shorthand syntax for defining anonymous functions. They are defined using the => operator, and are often used for simple, one-line functions. For example:

    ```
    const myFunction = (x) => x + 1;
    ```