1. console.log(10 + 10);

    - **Output:** 20
    - **Explanation:** Both operands are numbers, so they are added.

2. console.log(10 + "10");

    - **Output:** "1010"
    - **Explanation:** When a number is added to a string, the number is converted to a string and concatenated.

3. console.log(10 + +"10");

    - **Output:** 20
    - **Explanation:** The unary + operator converts the string "10" to a number, so it becomes 10 + 10.

4. console.log(10+"10"+10);

    - **Output:** "101010"
    - **Explanation:** 10+"10" results in "1010" (string concatenation), and then "1010"+10 results in "101010".

5. console.log(10+ +"10" + 10);

    - **Output:** 30
    - **Explanation:** +"10" converts the string "10" to a number 10, so it becomes 10 + 10 + 10.

6. console.log(10 - "2");

    - **Output:** 8
    - **Explanation:** The string "2" is converted to a number 2, so it becomes 10 - 2.

7. console.log(10 - "2" - "8");

    - **Output:** 0
    - **Explanation:** Both strings "2" and "8" are converted to numbers, so it becomes 10 - 2 - 8.

8. console.log(10+"2" - "2");

    - **Output:** 100
    - **Explanation:** 10+"2" results in "102" (string concatenation). Then, "102" - "2" converts both to numbers and performs subtraction: 102 - 2.

9. console.log(10>9>8);

    - **Output:** false
    - **Explanation:** 10 > 9 is true. Then true is coerced to 1, so 1 > 8 is false.

10. console.log(10 * "10");

    - **Output:** 100
    - **Explanation:** The string "10" is converted to a number 10, so it becomes 10 * 10.

11. console.log(100 / "100");

    - **Output:** 1
    - **Explanation:** The string "100" is converted to a number 100, so it becomes 100 / 100.

12. console.log(100/"0");

    - **Output:** Infinity
    - **Explanation:** Division by zero results in Infinity in JavaScript.

13. console.log(100 + +"100" - "100" * "100");

    - **Output:** -9800
    - **Explanation:** +"100" is 100, "100" * "100" is 10000, so it becomes 100 + 100 - 10000.

14. console.log(1 == "1");

    - **Output:** true
    - **Explanation:** == performs type coercion, so "1" is converted to 1 and 1 == 1.

15. console.log(1 === "1");

    - **Output:** false
    - **Explanation:** === checks for strict equality without type coercion, so a number is not equal to a string.

16. console.log(1 == "one");

    - **Output:** false
    - **Explanation:** The string "one" cannot be converted to a number, so 1 == NaN is false.

17. console.log(1 === "one");

- **Output:** false
- **Explanation:** Strict equality between a number and a string is false.

18. console.log(1 + true);

    - **Output:** 2
    - **Explanation:** true is coerced to 1, so 1 + 1.

19. console.log(1 - true);

    - **Output:** 0
    - **Explanation:** true is coerced to 1, so 1 - 1.

20. console.log(1 + true - false);

    - **Output:** 2
    - **Explanation:** true is 1 and false is 0, so it becomes 1 + 1 - 0.

21. console.log("1" + true);

    - **Output:** "1true"
    - **Explanation:** The number is converted to a string and concatenated.

22. console.log(+"1" + true);

    - **Output:** 2
    - **Explanation:** +"1" converts the string to 1, so it becomes 1 + 1.

23. console.log(undefined == undefined);

    - **Output:** true
    - **Explanation:** Both sides are undefined.

24. console.log(undefined === undefined);

    - **Output:** true
    - **Explanation:** Both sides are undefined and of the same type.

25. console.log(null == null);

    - **Output:** true
    - **Explanation:** Both sides are null.

26. console.log(null === null);

    - **Output:** true
    - **Explanation:** Both sides are null and of the same type.

27. console.log(undefined == null);

- **Output:** true
- **Explanation:** undefined and null are loosely equal.

28. console.log(undefined === null);

- **Output:** false
- **Explanation:** undefined and null are not strictly equal.

29. console.log(2+NaN);

- **Output:** NaN
- **Explanation:** Any operation with NaN results in NaN.

30. console.log("2"+NaN);

- **Output:** "2NaN"
- **Explanation:** NaN is converted to a string and concatenated.

31. console.log("2"+undefined);

- **Output:** "2undefined"
- **Explanation:** undefined is converted to a string and concatenated.

32. console.log(2+undefined);

- **Output:** NaN
- **Explanation:** undefined is coerced to NaN, so 2 + NaN is NaN.

33. console.log(typeof "123");

- **Output:** "string"
- **Explanation:** "123" is a string.

34. console.log(typeof 2);

- **Output:** "number"
- **Explanation:** 2 is a number.

35. console.log(typeof true);

- **Output:** "boolean"
- **Explanation:** true is a boolean.

36. console.log(typeof undefined);

- **Output:** "undefined"
- **Explanation:** undefined is of type undefined.

37. console.log(typeof null);

- **Output:** "object"
- **Explanation:** null is historically an object type in JavaScript.

38. console.log(typeof []);

- **Output:** "object"
- **Explanation:** Arrays are objects in JavaScript.

39. console.log(typeof 1n);

- **Output:** "bigint"
- **Explanation:** 1n is a BigInt.

40. console.log(typeof 1n+2n);

- **Output:** "bigint2n"
- **Explanation:** typeof 1n is "bigint", and "bigint" + 2n results in a concatenation.

41. console.log(typeof 1+2n);

- **Output:** "number2n"
- **Explanation:** typeof 1 is "number", and "number" + 2n results in a concatenation.

42. console.log(typeof 1/1n);

- **Output:** NaN
- **Explanation:** typeof 1 is "number", and division by a BigInt is not allowed, resulting in NaN.

1. What is the value of x after the operation: x = 5 + 3 * 2;?

let x = 5 + 3 * 2;

console.log(`x = ${x}`);

Output: x = 11

Explanation:

- The multiplication 3 * 2 is performed first, resulting in 6.
- Then the addition 5 + 6 is performed.
- Thus, the value of x is 11.

2. What is the value of y after the operation: y = 12 - 4 / 2;?

```
let y = 12 - 4 / 2;

console.log(`y = ${y}`);
```

Output: y = 10

Explanation:

- The division 4 / 2 is performed first, resulting in 2.
- Then the subtraction 12 - 2 is performed.
- Thus, the value of y is 10.

3. What is the value of z after the operation: z = 7 + 2 * 3 - 1;?

```
let z = 7 + 2 * 3 - 1;

console.log(`z = ${z}`);
```

Output: z = 12

Explanation:

- The multiplication 2 * 3 is performed first, resulting in 6.
- Then the addition 7 + 6 is performed, resulting in 13.
- Finally, the subtraction 13 - 1 is performed.
- Thus, the value of z is 12.

4. What is the value of a after the operation: a = 9 % 3 + 2;?

```
let a = 9 % 3 + 2;

console.log(`a = ${a}`);
```

Output: a = 2

Explanation:

- The modulus operation 9 % 3 is performed first, resulting in 0 (since 9 is exactly divisible by 3).
- Then the addition 0 + 2 is performed.

- Thus, the value of a is 2.

5. What is the value of b after the operation: b = 15 / 3 * 2;?

```
let b = 15 / 3 * 2;

console.log(`b = ${b}`);
```

Output: b = 10

Explanation:

- The division 15 / 3 is performed first, resulting in 5.
- Then the multiplication 5 * 2 is performed.
- Thus, the value of b is 10.

6. What is the value of c after the operation: c = 24 >> 2;?

```
let c = 24 >> 2;

console.log(`c = ${c}`);
```

Output: c = 6

Explanation:

- The >> operator is a bitwise right shift. It shifts the bits of the number to the right by the specified number of positions.
- 24 in binary is 11000.
- Shifting right by 2 positions gives 0110, which is 6 in decimal.
- Thus, the value of c is 6.

7. What is the value of d after the operation: d = 17 & 3;?

```
let d = 17 & 3;

console.log(`d = ${d}`);
```

Output: d = 1

Explanation:

- The & operator is a bitwise AND. It compares each bit of the first operand to the corresponding bit of the second operand. If both bits are 1, the corresponding result bit is set to 1.
- 17 in binary is 10001.

- 3 in binary is 00011.
- Performing bitwise AND: 10001 & 00011 = 00001, which is 1 in decimal.
- Thus, the value of d is 1.

8. What is the value of e after the operation: e = 28 ^ 2;?

```
let e = 28 ^ 2;

console.log(`e = ${e}`);
```

Output: e = 30

Explanation:

- The ^ operator is a bitwise XOR. It compares each bit of the first operand to the corresponding bit of the second operand. If the bits are different, the corresponding result bit is set to 1.
- 28 in binary is 11100.
- 2 in binary is 00010.
- Performing bitwise XOR: 11100 ^ 00010 = 11110, which is 30 in decimal.
- Thus, the value of e is 30.

9. What is the value of f after the operation: f = 11 + 3 << 2;?

```
// Question 9: f = 11 + 3 << 2

let f = 11 + 3 << 2;

console.log(`f = ${f}`);
```

Output: f = 56

Explanation:

- Addition is performed before the bitwise left shift.
- 11 + 3 is calculated first, resulting in 14.
- 14 in binary is 1110.
- Shifting left by 2 positions gives 111000, which is 56 in decimal.
- Thus, the value of f is 56.

10. What is the value of g after the operation: g = 25 - 5 | 3;?

```
let g = 25 - 5 | 3;

console.log(`g = ${g}`);
```

Output: g = 23

Explanation:

- Subtraction is performed before the bitwise OR.
- 25 - 5 is calculated first, resulting in 20.
- 20 in binary is 10100.
- 3 in binary is 00011.
- Performing bitwise OR: 10100 | 00011 = 10111, which is 23 in decimal.
- Thus, the value of g is 23.

1. What is the value of granted after the operation?

let username = "admin";

let password = "password";

let granted = (username === "admin" && password === "password") ? true : false;

Output: granted = true;

Explanation:

- The condition inside the parentheses (username === "admin" && password === "password") checks if both username is "admin" and password is "password".
- Since both conditions are true, the entire expression evaluates to true.
- The ternary operator ? then sets granted to true.

2. What is the value of message after the operation?

let username = "user";

let password = "wrongpassword";

let message = (username === "admin" && password === "password") ? "Login successful!" : "Invalid credentials.";

Output: message = "Invalid credentials.";

Explanation:

- The condition (username === "admin" && password === "password") checks if both username is "admin" and password is "password".
- Since username is "user" and password is "wrongpassword", both conditions are false.
- The ternary operator ? then sets message to "Invalid credentials.".

3. What is the value of access after the operation?

let username = "admin";

let password = "password";

let access = (username === "admin" || password === "password") ? "Granted" : "Denied";

Output: access = "Granted";

Explanation:

- The condition (username === "admin" || password === "password") checks if either username is "admin" or password is "password".
- Since both conditions are true, the entire expression evaluates to true.
- The ternary operator ? then sets access to "Granted".

4. What is the value of status after the operation?

let username = "";

let password = "password";

let status = (username !== "" && password === "password") ? "Logged in" : "Please enter username and password";

Output: status = "Please enter username and password";

Explanation:

- The condition (username !== "" && password === "password") checks if username is not an empty string and password is "password".
- Since username is an empty string, the first condition is false.
- The ternary operator ? then sets status to "Please enter username and password".

5. What is the value of authenticated after the operation?

let username = "admin";

let password = "wrongpassword";

let authenticated = (username === "admin" && password === "password") ? true : false;

Output: authenticated = false;

Explanation:

- The condition (username === "admin" && password === "password") checks if both username is "admin" and password is "password".
- Since username is "admin" but password is "wrongpassword", the second condition is false.
- The ternary operator ? then sets authenticated to false.

1. What is the value of name after the operation?

```
let user = { name: "John" };

let name = user?.name ?? "Unknown";

console.log(name);
```

Output: "John"

Explanation:

- user?.name uses optional chaining to safely access the name property of user. If user is null or undefined, it would return undefined instead of throwing an error.
- ?? is the nullish coalescing operator which returns the right-hand side operand when the left-hand side is null or undefined.

Since user.name is "John", the resulting value of name is "John".

2. What is the value of price after the operation?

```
let product = { price: null };

let price = product?.price ?? "N/A";

console.log(price);
```

Output: "N/A"

Explanation:

- product?.price attempts to access the price property of product. If product is null or undefined, it returns undefined.
- ?? operator checks if the left-hand side is null or undefined.

Since product.price is null, the resulting value of price is "N/A".

3. What is the value of address after the operation?

```
let customer = { address: { street: "123 Main St" } };

let address = customer?.address?.street ?? "Not available";

console.log(address);
```

Output: "123 Main St"

Explanation:

- customer?.address?.street safely accesses the street property. If customer or address is null or undefined, it returns undefined.
- ?? operator returns the right-hand side if the left-hand side is null or undefined.

Since customer.address.street is "123 Main St", the resulting value of address is "123 Main St".

4. What is the value of phone after the operation?

```
let contact = { phone: null };

let phone = contact?.phone ?? "Not provided";

console.log(phone);
```

Output: "Not provided"

Explanation:

- contact?.phone attempts to access the phone property. If contact is null or undefined, it returns undefined.
- ?? operator checks if the left-hand side is null or undefined.

Since contact.phone is null, the resulting value of phone is "Not provided".

5. What is the value of description after the operation?

```
let item = { description: "" };

let description = item?.description ?? "No description available";

console.log(description);
```

Output: ""

Explanation:

- item?.description attempts to access the description property. If item is null or undefined, it returns undefined.
- ?? operator checks if the left-hand side is null or undefined.

Since item.description is an empty string "" (which is neither null nor undefined), the resulting value of description is "".