1. console.log(10 + 10);
Ans:20

2. console.log(10 + "10");
Ans:1010 typeof ans is String

3. console.log(10 + +"10");
Ans:20
The +"10" part tries to convert the string "10" to a number. The unary plus operator (+)
converts its operand to a number. Since "10" can be converted to the number 10.

4. console.log(10+"10"+10);
Ans:101010
In javascript,when we use + operator for numbers and string it converts all numbers into strings and concatenates it.

5. console.log(10+ +"10" + 10);
Ans:30
The +"10" part tries to convert the string "10" to a number. The unary plus operator (+)
converts its operand to a number. Since "10" can be converted to the number 10.

6. console.log(10 - "2");
Ans:8
In javascript, when we use arithmetic operator(except + operator,because it can add numbers and
also concatenate two strings) between string and number String is type converted automatically into number.

7. console.log(10 - "2" - "8");
Ans:0
In javascript, when we use arithmetic operator between string and number String is type
converted automatically.

8. console.log(10+"2" - "2");
Ans:100
Here, 10+"2" will be "102" ,after 102 is converted into number because of "-2".

9. console.log(10>9>8);
Ans:False
Here,10>9 gives True,True means 1,so next it evaluates 1>8 and it returns False.

10. console.log(10 * "10");
Ans:100
In javascript, when we use arithmetic operator(except + operator,because it can add numbers and also concatenate two strings) between string and number String is type
converted automatically into number.

11. console.log(100 / "100");
Ans:1

In javascript, when we use arithmetic operator(except + operator,because it can add numbers and also concatenate two strings) between string and number String is type
converted automatically into number.

12. console.log(100/"0");
Ans:Infinity
In javascript, when we use arithmetic operator(except + operator,because it can add numbers and also concatenate two strings) between string and number String is type
converted automatically into number.

13. console.log(100 + +"100" - "100" * "100");
Ans:-9800
+"100" converts into 100 ,after 100 + +"100" becomes 200 then "100" * "100" value is 10000,after evaluating we get 200-10000=-9800

14. console.log(1 == "1");
Ans:True
In Javascript,when we use "==" only the values are checked whether they are equal are not.

15. console.log(1 === "1");
Ans:False
In Javascript,when we use "===" it strictly checks the values and datatype and gives the result.

16. console.log(1 == "one");
Ans:False
Here,the values are not same, so it gives False as result.

17. console.log(1 === "one");
Ans:False
Here,the values and datatypes are not same, so it gives False as result.

18. console.log(1+true);
Ans:2
Here,true is converted into 1 so the result is 2.

19. console.log(1 - true);
Ans:0
Here,true is converted into 1 so the result is 0.

20. console.log(1 + true - false);
Ans:2
Here,true is 1 and false is 0,hence the result is 2.

21. console.log("1" + true);
Ans:1true
Since 1 is a string , true will be concatenated to 1.

22. console.log(+"1" + true);
Ans:2
+"1" will become 1 and the answer will be 2.
23. console.log(undefined == undefined);
Ans:True
In JavaScript, when you use the == operator for comparison, it performs type coercion, which means it

tries to convert both operands to a common type before making the comparison.

When comparing undefined with undefined, they are of the same type (undefined) and hold the
same value (which is essentially "no value"). So, during comparison, they are considered equal.

24. console.log(undefined === undefined);
Ans:True
When comparing undefined with undefined, they are of the same type (undefined) and hold the same
value (which is essentially "no value"). So, during comparison, they are considered equal.

25. console.log(null == null);
Ans:True
When comparing null with null, they are of the same type (null) and hold the same
value (which is essentially "no value"). So, during comparison, they are considered equal.

26. console.log(null === null);
Ans:True
When comparing null with null, they are of the same type (null) and hold the same
value (which is essentially "no value"). So, during comparison, they are considered equal.

27. console.log(undefined == null);
Ans:True
when comparing null and undefined with "==", they have same value(no value),so the result is true.

28. console.log(undefined === null);
Ans: False
when comparing null and undefined with "===", they have same value(no value) but they are of different datatype
so result is false.

29. console.log(2+NaN);
Ans:NaN
In JavaScript, performing arithmetic operations involving NaN (Not-a-Number) results in NaN.

When you try to add a number to NaN, the result will always be NaN. This behavior is consistent
with the IEEE 754 standard for floating point arithmetic, which JavaScript adheres to.

30. console.log("2"+NaN);
Ans:2NaN
since 2 is a string, NaN is also taken as string and concatenated with it.

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

since 2 is a string, undefined is also taken as string and concatenated with it.

32. console.log(2+undefined);
Ans:NaN
When you perform arithmetic operations involving undefined in JavaScript, the result is NaN (Not-a-Number).

This behavior is because undefined represents a variable that has been declared but hasn't been assigned a
value. When you try to perform arithmetic operations with such a value, JavaScript treats it as if it were
NaN, indicating that the operation cannot be performed sensibly.

33. console.log(typeof "123");
Ans:String
In javascript,typeof is used to determine the datatype of given value.

34. console.log(typeof 2);
Ans:number
In javascript,typeof is used to determine the datatype of given value.

35. console.log(typeof true);
Ans:boolean
In javascript,typeof is used to determine the datatype of given value.

36. console.log(typeof undefined);
Ans:undefined
In javascript,typeof is used to determine the datatype of given value.

37. console.log(typeof null);
Ans:object
In javascript,typeof is used to determine the datatype of given value.

38. console.log(typeof []);
Ans:Object
In javascript,typeof is used to determine the datatype of given value.

39. console.log(typeof 1n);
Ans:bigint
In javascript,typeof is used to determine the datatype of given value.

40. console.log(typeof 1n+2n);
Ans:bigint2
In JavaScript, when you perform an operation involving BigInts, the result is
always a BigInt. Therefore, when you write typeof 1n + 2n, JavaScript evaluates
typeof 1n first, which results in "bigint", and then adds 2n to it.

So, typeof 1n + 2n is interpreted as "bigint" + 2n, resulting in the concatenation of the string "bigint" with the string representation of the BigInt 2n, yielding "bigint2".

41. console.log(typeof 1+2n);
Ans:number2
In JavaScript, the typeof operator is a unary operator with a higher precedence

than the addition operator +. Therefore, JavaScript first evaluates typeof 1,
which returns "number", and then tries to concatenate this string with the BigInt 2n.

So, typeof 1 + 2n is interpreted as "number" + 2n, which results in the concatenation
of the string "number" with the string representation of the BigInt 2n, yielding "number2".

42. console.log(typeof 1/1n);
Ans:TypeError: Cannot mix BigInt and other types, use explicit conversions
In this case, when you write typeof 1/1n, JavaScript interprets it as (typeof 1) / 1n.
The typeof operator is unary and has higher precedence than the division operator /.

The typeof operator returns a string indicating the type of the operand, in this case,
the number 1. Then, it attempts to divide this string by the BigInt 1n, which results
in a TypeError because you cannot divide a string by a BigInt.

---

1. What is the value of x after the operation: x = 5 + 3 * 2;?
Ans:11
5 + 3 * 2
5+6
11

2. What is the value of y after the operation: y = 12 - 4 / 2;?
Ans:10
12-4/2
12-2
10

3. What is the value of z after the operation: z = 7 + 2 * 3 - 1;?
Ans:12
7+6-1
7+5
12

4. What is the value of a after the operation: a = 9 % 3 + 2;?
Ans:2
9%3+2
0+2
2

5. What is the value of b after the operation: b = 15 / 3 * 2;?
Ans:10
15/3*2
5*2
10

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

Ans:6
24 in binary 11000
11000>>2 = 110 =6

7. What is the value of d after the operation: d = 17 & 3;?
Ans:1
17 in binary 10001
          & 00011(3 in binary)
            00001 =1

8. What is the value of e after the operation: e = 28 ^ 2;?
Ans:30

9. What is the value of f after the operation: f = 11 + 3 << 2;?
Ans:56
11+3<<2
14<<2
1110<<2
111000 in decimal is 56

10.What is the value of g after the operation: g = 25 - 5 | 3;?
Ans:23
25-5|3
20|3
 10100
|00011

 10111 in decimal 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;
```

Ans:true
username === "admin" is true , password === "password" is true, true &&
true = 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.";
```

Ans:Invalid credentials.
username === "admin" is false , password === "password" is false, false
&& false = false,so it prints 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";
```

Ans:Granted
username === "admin" is true, password === "password" is true,true || true is true, so it prints 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";
```

Ans:Please enter username and password
username !== "" is false ,password === "password" is true , false && true is false, so it prints 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;
```

Ans:false
username === "admin" is true, password === "password" is false, true && false is false,so it prints false.

---

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

```
let user = { name: "John" };
let name = user?.name ?? "Unknown";
```

Ans:John
user?.name gives john,if name is null we get unknown,since it is not null we get john.

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

```
let product = { price: null };
let price = product?.price ?? "N/A";
```

Ans:N/A
product?.price is null, since it is null , we get N/A as output.

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

```
let customer = { address: { street: "123 Main St" } };
let address = customer?.address?.street ?? "Not available";
```

Ans:123 Main St
customer?.address?.street gives the value 123 main st and it is printed

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

```
let contact = { phone: null };
let phone = contact?.phone ?? "Not provided";
```

Ans:Not provided
contact?.phone is null, since we used null coalescing operator the default value will be printed,i.e Not provided.

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

```
let item = { description: "" };
let description = item?.description ?? "No description available";
```

Ans:empty String
item?.description gives emptystring as output, the default value will only be printed if the value is null, since empty string is not null it is printed.