Day - 13

JavaScript Output Based Question

github → rajeshjha2000

(61)
```
const name = "Happy Coding";
console.log(!typeof name === 'object');
console.log(!typeof name === 'string);
```

Ans:- 1. Operator Precedence ⟶

- In JS, the precedence of the '!' (logical NOT) operator is higher than the 'typeof' operator. However, 'typeof name' is treated as a single operation and is evaluated first.
- After 'type of name' is evaluated, the result ('string') is then subjected to the '!' operator, and finally, the comparison is made.

2. '!typeof name === 'object'; ⟶

- 'typeof name' evaluates to 'string'.
- '!typeof name' is evaluated as '!'string''. In JS, any non empty string is considered truthy, so '!'string'' becomes 'false'.
- Now, the expression is 'false === 'object''.
- 'false' is a boolean, and ''object'' is a string, so they are not strictly equal.

3. Second line: '!typeof name === 'string'; ⟶

- Again, 'typeof name' evaluates to "string".
- '!typeof name' is evaluated as '!'string'', which again becomes 'false'.

• Now, the expression is 'false === 'string''.
• As before, 'false' and "string'' are not strictly equal.

Therefore, the output will be ⟶
• false
• false

Note ⟶ console.log(!typeof name === false);
• true

(62) Const name = "rajesh";
const age = 24;

console.log(isNaN(name));
console.log(isNaN(age));

Ans:- 1. `isNaN()' Function ⟶
• The 'isNaN()' function checks whether a value is `NaN'
(Not-a-Number).
• However, before the check, the value is first coerced to a number. If the coercion results in a value that is 'NaN', then 'isNaN()' returns `true';
Otherwise, it returns `false'.

2. `isNaN(name)'; ⟶
• The variable 'name' holds the string "rajesh".
• when, ~~[illegible]~~ "rajesh" is coerced to a number, it results in `NaN' because "rajesh" is not a numeric value.
• Therefore, 'isNaN("rajesh")' returns 'true'.

3. 'isNaN(age)' : →

- The variable 'age' holds the number '24'.
- Since '24' is already a number 'isNaN(24)' returns 'false'.

Therefore, the output will be →

- true
- false

(63) ~~let person = {name : "[illegible]"~~ };

```
let data = "code";
console.log(typeof !data);
```

Ans) • '!data' evaluates to 'false' because "code" is truthy, and the logical NOT operator negate it.

- Now, 'typeof !data' is equivalent to 'typeof false'
- 'typeof false' returns the string ' "boolean" '.

Therefore, the output will be →

- "boolean"

(64)
```
console.log(parseInt('10+2'));
console.log(parseInt('7FM'));
```

Ans) 1. 'parseInt()' Function →

- The 'parseInt()' Function parses a string and returns an Integer.

- It reads the string from left to right and stops parsing when it encounters a character that isn't part of a valid number.

- If the first character in the string can't be converted to a number, 'parseInt()' returns 'NaN'.

2. First line: 'parseInt('10+2')': ⟶
- 'parseInt()' starts parsing the string '10+2' from left to right.
- It successfully parses the initial part '10' as an integer
- The parsing stops when it encounters the '+' symbol because '+' is not a valid numeric character for parsing an integer.
- Therefore, 'parseInt('10+2')' returns '10'.

3. Second line: 'parseInt('7FM')' ⟶
- The parsing stops when it encounters the 'F' character because it's not a valid numeric character.

Therefore, the output will be ⟶
- 10
- 7

(65)
```
[1,2,3].map(num => {
  if (num>0) return;
  return num×2;
});
```

1. 'map()' method →

- The 'map()' method creates a new array populated with the result of calling a provided function on every element in the calling array.
- The provided function takes each element ('num' in this case) and returns a value that will be placed in new array.

2. Logic Inside 'map()' →

- The function inside 'map()' checks if the number ('num') is greater than '0'.
- If 'num >0', the function returns 'undefined' (since 'return' without any value results in 'undefined'),

Therefore, the output will be →

- [undefined, undefined, undefined]

For more questions, Visit →

gitHub → rajeshjha2000