Day-15

JavaScript Output Based Questions

gitHub → rajeshjha2000

Q.71> function Sum (a=5, b=7){

console.log(a+b);

}

Sum (null, 20) ;

Ans> 1. Function Definition ('sum(a=5, b=7)') :→

- The 'Sum' function is defined with two parameters, 'a' & 'b'.
- Both parameters have default values : 'a' defaults to '5' and 'b' defaults to '7'.

2. Function Call ('sum(null, 20)') →

- We call the function with two arguments : 'null' for 'a' and '20' for 'b'.
- Since 'null' is explicitly passed as the first argument, it overrides the default value of 'a'.
- The value '20' is passed for 'b', so it overrides the default value of '7'.

3. Behavior of 'null' →

- In JS, 'null' is a falsy value, but it is still a valid value.
- When we pass 'null' as an argument, it does not trigger the default value ; instead 'null' is used directly.
- Null is coerced to '0'.

null + 20 = 0 + 20

Therefore, the output will be →

- 20

(72) let newlist = [2,3].push(4);
console.log(newlist.push(5));

Ans:> 1. let newlist = [2,3].push(4) →

- The 'push' method adds one or more elements to the end of an array and returns the new length of the array.
- In this case, '[2,3].push(4)' will add '4' to the array '[2,3]', resulting in '[2,3,4]'.
- The 'push' method returns the new length of array, which is '3'.
- Therefore, 'newlist' will not be an array, it will be a number with the value '3'.

2. console.log(newlist.push(5)) →

- Since 'newlist' is a number ('3'), not an array, attempting to call 'push' on it will result in an error.

Therefore, the output will be →

- TypeError : newList.push is not a function

Note → The reason why the result of the 'push()' method became a number, rather than remaining an array, lies in how the 'push()' method is designed to work in JS.

- The '.push()' method adds one or more elements to the end of an array.

- However, the '.push()' method does not return the array itself after adding the element. Instead, it returns the new length of the array after the elements have been added.

- Correct method → let newlist = [2,3];
  newList.push(4);
  console.log(newlist); //output→ [2,3,4]

(73) function getItems (list, ...args, moreItem) {
  return [...list, ... args, moreItems]
}
getItems (["berry", "apple"], "pear", "kiwi");

Ans:- • The function 'getItems' is defined with parameters 'list', '...args', and 'moreItem'.

• '...args' is a rest parameter which gathers all remaining arguments into an array.

• In JS, the rest parameter '...args' must be the last parameter in a function's parameter list. However, in this function, 'moreItem' comes after '...args', which is not allowed.

Therefore, the output will be →

• Syntax Error: Rest element must be last element

(74)
```
function nums (a,b){
if (a>b) console.log ('a is large');
else console.log ('b is large');
return
 a+b;
}
console.log (nums (4,2));
console.log(nums (1,2)) ;
```

A-> Function Call 'nums (4,2)' ⟶

- '4>2' is 'true', so it prints 'a is large'.
- The function returns '4+2', which is '6'.

Function Call 'nums (1,2)' ⟶

- '1>2' is 'false', so it prints 'b is large'.
- The function returns '1+2', which is '3'.

Therefore, the output will be ⟶

- a is large
- 6
- b is large
- 3

(75)
```
class Person {
  constructor() {
  this.name = 'Rajesh';
  }
}

Person = class AnotherPerson {
  constructor() {
   this.name = 'Jha';
  }
};

const member = new Person();
console.log(member.name);
```

Ans> In this code, the 'person' class is reassigned to new class 'AnotherPerson' before creating an instance. This reassignment effectively replaces the original 'person' class with 'AnotherPerson'.

The original 'Person' class is no longer accessible, and 'Person' now refers to 'AnotherPerson'.

→ Creating an Instance of 'Person' →

```
const member = new Person();
```

- Since 'Person' is now 'AnotherPerson', this line creates an instance of 'AnotherPerson'.

Therefore, the output will be →

- Jha