Day-16

JavaScript Output Based Question

github → rajeshjha2000

(76) let num = 1;
Const list = ['♡', '☺', '👍'];

num = num+1;
console.log(list[num]);

Ans:> 1. Variable Initialization →
- 'num' is initialized to '1'.
- 'list' is an array containing three elements: '♡', '☺', '👍'.

2. Updating 'num' →
- 'num' is incremented by '1', so 'num' becomes '2'.

3. Accessing Array Element →
- Arrays in JS are zero-indexed, so the indices are:
  - 'list[0]' is '♡'
  - 'list[1]' is '☺'
  - 'list[2]' is '👍'

Therefore, the output will be →
- 👍

(77) let randomValue = {name: "Rajesh"}
randomValue = 24

```
if(!typeof randomValue === "string") {
    console.log("It's not a string!")
}
  else {
  console.log("Hey it's a string!")
}
```

Ans:) 1. Variable Initialization and Reassignment →

- 'randomValue' is initially an object with a property 'name' set to "Rajesh".
- Then, 'randomValue' is reassigned to '24', so it is now a number.

2. Condition check →

- The 'typeof randomValue' returns "number", since 'randomValue' is now a number.
- The '!' operator only negates the immediate expression that follows it, which in this case is 'typeof randomValue'. However, the 'typeof' operator always returns a string, so '!typeof randomValue' will always be 'false', since any non-empty string is truthy and negating it will be 'false'.
- The expression '!typeof randomValue === "String" is evaluated as:
  - 'false === "String"', which is 'false'.
- Thus, the 'else' block will always execute regardless of the actual type of 'randomValue'.

Therefore, the output will be →

- Yay it's a String!

(78) const emoji = ['♡', '☺', '👍'];

```
emoji.slice(0,1);
emoji.splice(0,1);
emoji.unshift('☹');

console.log(emoji);
```

Ans> 1. 'slice' Operation →

```
emoji.slice(0,1);
```

- The 'slite' method returns a shallow copy of a portion of an array into a new array. In this case, it returns '['♡']' but it does not modify the original 'emoji' array.
- After this operation, 'emoji' remains '['♡', '☺', '👍']'.

2. 'Splice' Operation →

```
emoji.Splice(0,1);
```

- The 'splice' method changes the contents of an array by removing or replacing existing elements and/or adding new elements in place.
- 'emoji.Splice(0,1)' removes the first element from the 'emoji' array ('♡'), modifying 'emoji' to '['☺', '👍']'.

3. 'Unshift' operation →

- The 'unshift' method adds one or more elements to the beginning of an array and returns the new length of the array.

Therefore, the output will be →

→ ['☹', '☺', '👍']

(79)

```
let count = 0;
const nums = [0, 1, 2, 3];

nums.forEach (num => {
  if (num) {
    count += 1;
  }
});

console.log (count)
```

Ans.) 1. 'forEach' loop →

- The 'forEach' method iterates over each element in the 'nums' array.
- During each iteration, the code checks whether 'num' is truthy (i.e., 'if (num)').
- If the condition is true 'count' is incremented by '1'.

2. Iteration Details →

- First Iteration ('num = 0') →

→ '0' is falsy in JS, so 'if (num)' is 'false' and 'count' remains '0'.

- Second Iteration ('num = 1') →

→ '1' is truthy, so 'if (num)' is 'true', and 'count' is incremented to '1'.

- Third Iteration ('num = 2') →

→ '2' is truthy, so 'if (num)' is 'true', and 'count' is incremented to '2'.

- Fourth Iteration ('num = 3') →

→ '3' is truthy, so 'if (num)' is 'true', and 'count' is incremented to '3'.

Therefore, the output will be →

- 3

(80)
```
const person = {
  name : 'Rajesh',
  address : {
    city : 'Faridabad',
  },
};
Object.freeze(person);
person.name = null;
person.address.city = null;
console.log(person);
```

Ans:- 1. Freezing the Object → `Object.freeze(person);`

- It makes the 'person' object immutable. This means that the properties of 'person' can't be changed, added or deleted. However, only the top-level properties are frozen. The 'address' object, which is a nested object, is not deeply frozen.

2. Attempting to modify properties →

- `person.name = null`, This attempt to change 'name' will fail silently (without error) in non-strict mode, and 'name' remains 'Rajesh'.

- `person.address.city = null`, This attempt to change the 'city' property of the 'address' object will succeed because 'address' was not frozen. The 'city' property will be changed to 'null'.

Therefore, the output will be →

```
{
  name : 'Rajesh',
  address : {
    city : null
  }
}
```

For more question, visit

github → rajeshjha2000