Day-5

JavaScript Output Based Question

gitHub→rajeshjha2000

(21)
```
const person = {name: 'Lydia'};

    function sayHi(age) {
    return `${this.name} is ${age}`;
     }

    console.log(sayHi.call(person, 24));
    console.log(sayHi.bind(person, 24));
```

A→ Using Call method ⟶

- The 'call' method calls a function with a given 'this' value and arguments provided individually.
- Here, 'sayHi.call(person, 24)' calls 'sayHi' with 'this' set to 'person' and 'age' set to '24'.
- The result is '"Lydia is 24"'.

using bind method ⟶

- The 'bind' method creates a new function that, when called, has its 'this' value set to the provided value, with a given sequence of arguments.
- Here, 'sayHi.bind(person, 24)' creates a new function with 'this' set to 'person' and 'age' set to '24'.
- However, the new function is not called immediately, it needs to be invoked to get the result.
- Therefore, the output is a reference to the new function, not the result of the function call.

Scanned with CamScanner

Therefore, the output will be →

- "Lydia is 24".
- [Function : bound sayHi]

Note → console.log(sayHi.bind(person, 24)());

- The '( )' at the end immediately invokes the newly created bound function.
- The bound function is immediately invoked with the 'this' value set to 'person' and 'age' set to '24'. The function returns the string.

Therefore, the output will be →

- "Lydia is 24".

(22)
```
function sayHi () {
  return (() => 0) ();
  }
console.log(typeOf sayHi());
```

Ans. (() => 0) () →

- The arrow function '( ) => 0' is a function that takes no arguments and return '0'.
- It is immediately invoked, so the result is '0'.

- 'sayHi' →
- The 'sayHi' function returns the value '0'.

- 'typeOf' →
- The 'typeOf' operator is used to determine the type of the value return by 'sayHi'.
- Since 'sayHi' returns '0', the 'typeOf' operator will return '"number"'.

Scanned with CamScanner

Therefore, the output will be ⟶

- number

(23)
```
function sayHi () {
  return () => 0;
  }
Console.log(typeOf sayHi());
```

A→ . sayHi () ⟶

- When 'sayHi' is called, it returns the arrow function '() => 0'.

Therefore, the output will be ⟶

- function

(24)
```
console.log(typeof typeOf 1);
```

A→ • Inner 'typeOf' operation ⟶

- Since '1' is a number, 'typeof 1' evaluates to the string '"number"'.

- Outer 'typeOf' operation ⟶

- Now, we have 'typeOf "number"' because the result of 'typeOf 1' is '"number"'.

- Since '"number"' is a string, 'typeOf "number"' evaluates to '"String"'.

Therefore, the output will be ⟶

- String

Scanned with CamScanner

(25) const numbers = [1, 2, 3];
numbers[9] = 11;
console.log(numbers);

A→ • Array Def ⟶

• An array 'numbers' is defined with three elements: '[1,2,3]'.

• Setting an Element at Index 9 ⟶

• The array 'numbers' is updated by setting the element at index '9' to '11'.

• JS arrays are sparse, which means they can have gaps. When we set an element at an index that is beyond the current length of the array, it creates empty slots (holes) in the array.

Therefore, the output will be ⟶

• [1, 2, 3, <6 empty items>, 11]

Scanned with CamScanner