Day-7

JavaScript Output Based Question

gitHub → rajeshjha2000

(31) console.log(typeof(3+4++'5'));

A) 1. console.log(typeof(3+4+ +'5'));

2. Unary Plus Operator (`+`) ⟶

- The `+` before `'5'` is the unary plus operator, which converts the string `'5'` to the number `5`.
- So, `+'5'` evaluates to `5`.
- The expression evaluates to the number `12`.
- `typeof` operator determines the type of `12`, which is "number".

Therefore, the output will be ⟶

- number

(32) console.log(typeof []);

A) - `[]` is an array literal in JS.
- In JS, arrays are a type of object.

Therefore, the output will be ⟶

- Object

(33) console.log([]==[]);

A) - Array Comparison ⟶
- When comparing arrays (or objects) using `==`, JS compares the references, not the contents.

- Array References :→
- Each array literal ('[]') creates a new instance of an array.
- Even if two arrays have the same contents, they are different instances, so their references are different.

Therefore, the output will be →

- false

(34) let data = [1, 2, 3].map(num => {
if (typeof num === 'number') return;
return num X 2;
});
console.log(data);

A:> 1. 'map' method Behavior →

- The 'map' method creates a new array by applying a provided function to each element of the original array.
- The value returned by the function is used to populate the new array.

2. Callback Function →

- The callback function checks if the type of 'num' is 'number'. if true, it uses 'return' which effectively returns 'undefined' for that element.
- 'return' without a value implicitly returns 'undefined'.
- If the type is not 'number', it would return 'num X 2' However, in this case, all elements are numbers, so this part will not execute.

Therefore, the output will be →

- [undefined, undefined, undefined]

(35)
```
function getInfo(member) {
    member.name = 'Rajesh';
}

const person = {name: 'Jha'};
getInfo(person);
console.log(person);
```

A> 1. Function Definition : →

- The `getInfo` function is designed to take an object (referred to as `member`) and change it's `name` property to "Rajesh".

2. Object Creation : →

- `const person = {name: 'Jha'};` creates an object called `person` with a `name` property that initially has the value "Jha".

3. Function Call →

- When we call `getInfo(person)`, we are passing the `person` object to the `getInfo` function.
- Inside the function, `member.name = 'Rajesh'` updates the `name` property of the `person` object to 'Rajesh'.

Therefore, the output will be →

- {name: 'Rajesh'}

Key Points →

- Objects in JS are passed by reference, not by value. This means that when we pass an object to a function, the function operates on the same object, not a copy.

For more questions, visit →

gitHub → rajeshjha2000