Day - 11

JavaScript Output Based Question

gitHub → rajeshjha2000

(51) const data = {name: "Rajesh", age: 24, Skill: "JS"};
console.log(name);

Ans.> 1. Object Definition →

→ We have created an Object `data` with three properties: `name`, `age` and `skill`.

→ `data` is a local variable, containing an object with properties that can be accessed using dot notation or bracket notation.

2. Accessing the `name` Variable →

→ When we use `console.log(name);` JS looks for a variable named 'name' in the current scope.

→ However, 'name' is not declared as a separate variable in current scope; it's a property of the `data` Object.

→ However, 'name' is not declared as a separate variable in the current scope; it's a property of the `data` object.

Therefore, the output will be →

- Reference Error

correct method →

```
const {name} = data;
console.log(name);
```

Scanned with CamScanner

Note → • How to merge two Object ?

```
→ let data = {name: "Rajesh", age: 24}
→ let info = {city: "Faridabad", git: "rajeshjha2000"}
```

Ans:>
```
let Details = {...data, ...info}
console.log(Details);
```

(52)
```
function Human(fname, lname) {
  this.firstName = fname;
  this.lastName = lname;
}
const MrX = new Human("Mr.", "X");
const Rock = Human("The", "Rock");

console.log(MrX);
console.log(Rock);
```

Ans:> 1. The 'Human' Constructor Function →

• The 'Human' function is designed to act as constructor. It assigns the 'firstName' and 'lastName' properties to the newly created object when invoked with 'new' keyword.

2. Using 'new' with 'Human' →

• 'const MrX = new Human("Mr.", "X");'

• Here, 'MrX' is created as a new object with 'firstName' set to '"Mr."' and 'lastName' set to '"X"'.

Scanned with CamScanner

- This is the intended use of a constructor function, and `MrX` will correctly hold the properties `firstName: "Mr."` and `lastName: "X"`.

3. Calling `Human` without `new` ⟶
- `const Rock = Human("The", "Rock");`
- In this case, we are calling `Human` as a regular function, not as a constructor.
- When `Human` is called without `new`, `this` inside the function refers to the global object rather than a new instance.
- As a result, the properties `firstName` and `lastName` will be assigned to the global object (or `undefined` in strict mode)
- The function itself doesn't return anything explicitly, so `Rock` will be `undefined`.

Therefore, the output will be ⟶
- Human {firstName: 'Mr.', lastName: 'X'}
- Undefined

(53)
```
const name = 'Rajesh';
console.log(name());
```
A→ Since `name` is a string, attempting to call it as a function will result in a `TypeError`.

Therefore, the output will be ⟶
- TypeError : name is not a function

(54) const result = false || {} || null;
console.log(result);

Ans> 1. Logical OR ('||') operator →

- The '||' operator in JS returns the first truthy value it encounters or the last value if none are truthy.
- A value is considered "truthy" if it is not one of the following: 'false', '0', '""' (empty string), 'null', 'undefined', or 'NaN'.
- The evaluation is done from left to right.

2. Evaluation of expression →

- 'false' is falsy, so the evaluation continues.
- '{}' (an empty object) is truthy, so the evaluation stops, and '{}' is returned.

Therefore, the output will be →

- { }

(55) const result = [] || 0 || true;
console.log (result);

Ans> '[ ]' (an empty array) is truth value.

Therefore, the output will be →

- [ ]

H.W →(1) let data = {name: "Rajesh", age: 24};
let info = {city: "Faridabad"}
detail = {data, ... info}
console.log (detail);

(2) let data = {name: "Rajesh", skill: "JS"}
let info = {city: "Faridabad", skill: "Node"}
detail = {...data, ... info}
console.log (detail);

Scanned with CamScanner