

Guess the Output

1 Closure + Parameter Change

```
function outer(x) {  
  return function inner(y) {  
    console.log(x + y);  
  };  
}
```

```
let add5 = outer(5);  
add5(3);
```

? Output?

2 Higher-Order Function with Condition

```
function process(arr, fn) {  
  let result = [];  
  for (let i of arr) {  
    if (i % 2 === 0) {  
      result.push(fn(i));  
    }  
  }  
  return result;  
}
```

```
console.log(  
  process([1, 2, 3, 4], function (n) {  
    return n * 10;  
  })  
);
```

? Output?

3 IIFE + Closure

```
let count = 5;

(function () {
  let count = 10;
  return function () {
    console.log(count);
  };
})();
```

? Output?

4 Callback Execution Order

```
function first(cb) {
  console.log("First");
  cb();
}

function second() {
  console.log("Second");
}

first(second);
```

? Output?

5 **this** Inside Regular Function

```
let user = {
  name: "Amit",
  getName: function () {
    console.log(this.name);
  }
};

user.getName();
```

? Output?

6 **this** Lost in Callback ★

```
let user = {  
  name: "Ravi",  
  show: function () {  
    setTimeout(function () {  
      console.log(this.name);  
    }, 0);  
  }  
};
```

```
user.show();
```

? Output?

7 Fixing **this** using Arrow Function

```
let user = {  
  name: "Neha",  
  show: function () {  
    setTimeout(() => {  
      console.log(this.name);  
    }, 0);  
  }  
};
```

```
user.show();
```

? Output?

8 Hoisting + Block Scope

```
console.log(a);
```

```
let a = 10;
```

? Output?

9 Function vs Variable Hoisting

```
foo();

var foo = function () {
  console.log("Function Expression");
};

function foo() {
  console.log("Function Declaration");
}
```

? Output?

10 Closure Inside Loop (let vs var)

```
for (let i = 1; i <= 3; i++) {
  setTimeout(function () {
    console.log(i);
  }, 500);
}
```

? Output?

11 Higher-Order Function Returning Function

```
function greet(msg) {
  return function (name) {
    console.log(msg + " " + name);
  };
}

greet("Hello")("Suman");
```

? Output?

12 IIFE + Global Variable

```
var x = 1;
```

```
(function () {  
  x = 10;  
})();
```

```
console.log(x);
```

? Output?

13 Callback with Async Behavior

```
function task(cb) {  
  console.log("Task started");  
  setTimeout(cb, 0);  
}
```

```
task(function () {  
  console.log("Task finished");  
});
```

```
console.log("Outside");
```

? Output?

14 Array + Higher-Order Method

```
let arr = [10, 20, 30];
```

```
let sum = arr.reduce(function (acc, cur) {  
  return acc + cur;  
}, 0);
```

```
console.log(sum);
```

? Output?

15 Closure Memory Check

```
function parent() {  
  let msg = "Hello";  
  return function () {  
    msg = msg + " World";  
    console.log(msg);  
  };  
}
```

```
let fn = parent();  
fn();  
fn();
```

? Output?