1.INTRODUCTION TO JAVA SCRIPT

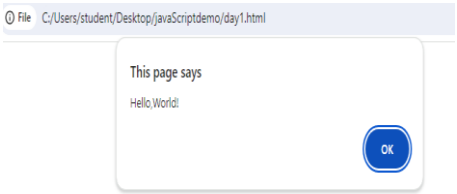
TASK1: Write a simple script that displays “Hello, World!” on the web page using an alert box.

CODE:

```
<script>

        alert("Hello,World!");
    </script>
```

OUTPUT:

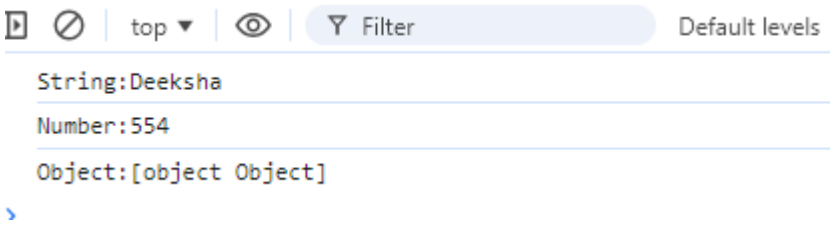


Task 2: Experiment with different data types in JavaScript (e.g., string, number, boolean) by declaring and logging them in the console

CODE:

```
<script>
       let s="Deeksha";
       console.log("String:"+s);
       let n = 554;
       console.log("Number:"+n);
       let myobject={
        name:"Deeksha",
        rollno:"T114"
       };
       console.log("Object:"+myobject);
    </script>
```

OUPUT:

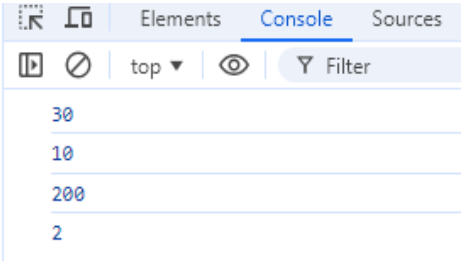


Task 3: Use the console to perform basic math operations like addition, subtraction, multiplication, and division.

CODE:

```
<script>
      let n1=20;
      let n2=10;
      console.log(n1+n2);
      console.log(+n1-n2);
      console.log(n1*n2);
      console.log(n1/n2);
    </script>
```

OUTPUT:

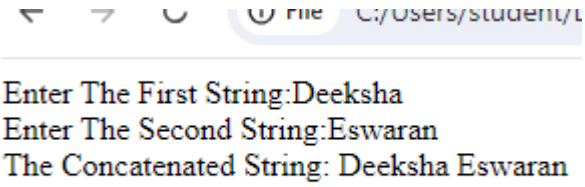


Task 4: Declare two strings and concatenate them using the + operator

CODE:

```
<script>
     var s1="Deeksha";
     var s2="Eswaran";
     document.write("Enter The First String:"+ s1+"<br>");
     document.write("Enter The Second  String:"+ s2+"<br>");
     document.write("The Concatenated String: "+s1 +" "+ s2+"<br>");
    </script>
```

OUTPUT:



Task 5: Use the typeof operator to check the data type of various variables

CODE:

```
<script>

     var M1=20;
     document.write(typeof(M1)+"<br>");
     var S1="Hello World";
     document.write(typeof(S1)+"<br>");
     var B1=762178912903780n;
```

```
    document.write(typeof(B1)+"<br>");
    var b1=true;
    document.write(typeof(b1)+"<br>");
    var n1=null;
    document.write(typeof(n1)+"<br>");
   </script>
```

OUTPUT:

number
string
bigint
boolean
object

2. CODE STRUCTURE:

Task 6: Write a multi-line JavaScript comment and a single-line comment. Explain the difference.

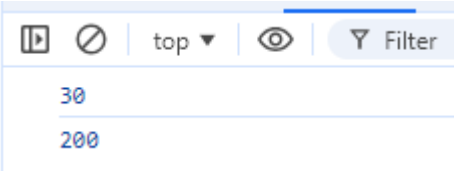
```
// This is a single-line comment
    let x = 10; // You can also add a single-line comment at the end of a line
of code
    /*
  This is a multi-line comment.
  It can span multiple lines.
  You can use it to write more detailed explanations or notes.
*/
let y = 20;
```

Task 7: Create a script with both semicolon-separated and not separated lines. Note any differences in behavior.

CODE:

```
let a=10;
let b=20;
console.log(a+b);
console.log(a*b)
</script>
```

OUTPUT:



Task 8: Use proper indentation to format a nested loop.

CODE:

```
let i,j;
for(i=0;i<2;i++){
    for(j=0;j<3;j++){
        document.write("Hello"+"<br>");
    }
    document.write("World"+"<br>");
}
```

OUTPUT:

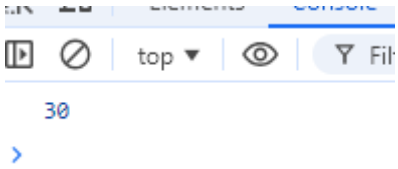
Hello
Hello
Hello
World
Hello
Hello
Hello
World

Task 9: Declare multiple variables in a single line.

CODE:

```
var a=10,b=20;
console.log(a+b);
</script>
```

OUTPUT:



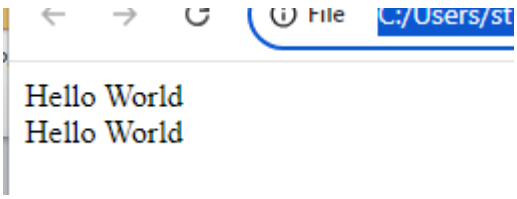
Task 10: Place a script tag at the top and bottom of an HTML document. Note any differences in behavior.

CODE:

```
<head>
    <script> document.write("Hello World"+ "<br>");</script>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script> document.write("Hello World");</script>
</body>
```

OUTPUT:



2: THE MODERN MODE, “USE STRICT”, VARIABLES

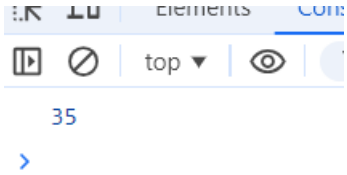
1. THE MODERN MODE, “USE STRICT”:

Task 11: Write a script without using “use strict” and try to assign a value to an undeclared variable. Note the result.

CODE:

```
a=35;
console.log(a);
```

OUTPUT:

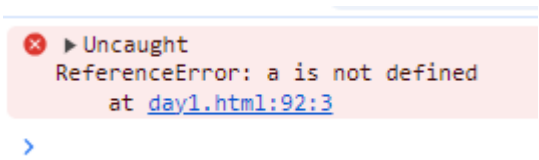


Task 12: Enable “use strict” mode and repeat the above action, noting the difference.

CODE:

```
    "use strict";
 a=35;
console.log(a);
</script>
```

OUTPUT:



Task 13: In “use strict” mode, try to delete a variable, function, or function parameter.

CODE:

```
"use strict";
var person = {
  firstName: "John",
  lastName: "henry",
  age: 24
};
delete person.age;
console.log(person);
</script>
```

OUTPUT:

```
top ▾   Filter   Default lev
▶ {firstName: 'John', lastName: 'henry'}
```

Task 14: Assign a value to an undeclared variable without “use strict” and then with “use strict”.

CODE:

```
"use strict"
 num=24;
 document.write(num+"<br>");
 number=24;
 document.write(number);
```

OUTPUT:

```
▶ Uncaught
  ReferenceError: num is not defined
      at day1.html:107:5
```

Task 15: Declare a variable with a reserved keyword in “use strict” mode.

CODE:

```
"use strict";

try {
    var let = 10;
} catch (e) {
    console.log("Error:", e.message);
}
```

OUTPUT:

```
Uncaught                                                    day1.html:114
SyntaxError: Unexpected strict mode reserved word (at day1.html:114:9)
```

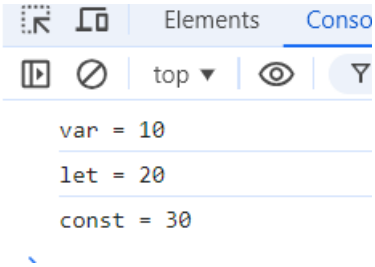
2. VARIABLES:

Task 16: Declare variables using let, const, and var. Discuss when each should be used.

CODE:

```
 var x=10;
console.log("var = " + x);
let y=20;
console.log("let = " + y);
const z=30;
console.log("const = " + z)
```

OUTPUT:



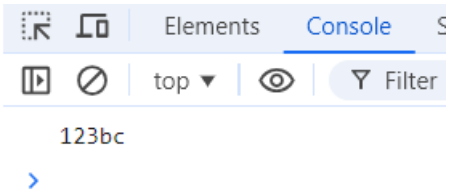
Task 17: Attempt to reassign a const variable and observe the result.

CODE:

```
const x="123bc";
const a="apple";
console.log(x);

</script>
```

OUTPUT:

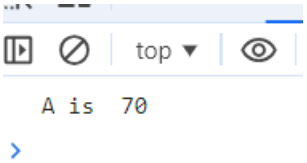


Task 18: Declare a variable without initializing it and print its value.

CODE:

```
var a=prompt("Enter the Value:");
console.log("A is  "+a);
```

OUTPUT:

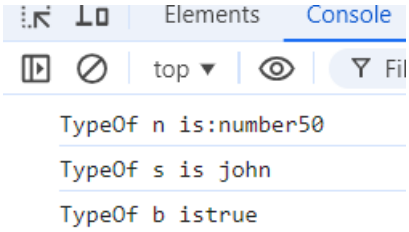


Task 19: Assign a number, string, and boolean value to a variable and print its type using typeof.

CODE:

```
var n=50;
var s="john";
var b=true;
console.log("TypeOf n is:"+typeof(n)+n);
console.log("TypeOf s is "+s);
console.log("TypeOf b is"+b);
```

OUTPUT:

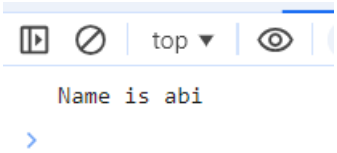


Task 20: Rename a variable and observe the outcome.

CODE:

```
var name="hari";
var name="abi";
console.log("Name is "+name);
```

OUTPUT:



3: Data types, Basic operators, maths

1. Data types:

Task 21: Create variables of different data types (e.g., string, number, boolean, null, undefined, object).

CODE:

```
var n="Henry";
console.log("String :"+ n);
var i=2456;
console.log("Number:"+ i);
var isStudent=true;
console.log("Boolean: "+ isStudent);
var emp=null;
console.log("Null: "+ emp);
var un;
console.log("undefined: "+ un);
```

OUTPUT:

```
String Henry
Number2456
Boolean true
Null null
undefined undefined
```

Task 22: Use the typeof operator to determine the type of various variables.

CODE:

```
var c = 2479487520n;
console.log("Type is : " +typeof(c));
```

OUTPUT:

```
Type is : bigint
>
```

Task 23: Declare a symbol and print its type.

CODE:

```
var s=Symbol("uniqueIdentifier");
console.log(" Typeof s is "+typeof(s));
</script>
```

OUTPUT:

```
Typeof s is symbol
>
```

Task 24: Assign the value null to a variable and check its type using typeof.

CODE:

```
var n=null;
console.log("Typeof n is "+typeof(n));
```

OUTPUT:

```
Typeof n is object
>
```

Task 25: Differentiate between declaring a variable using var and let in terms of scope.

CODE:

“var”:

Accessible throughout the entire function, even if declared inside a block.

“let”:

Only accessible within the block {} where it's declared.

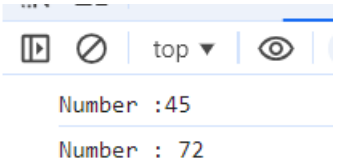
2. Basic operators, maths:

Task 26: Convert a string to a number using both implicit and explicit conversion.

CODE:

```
// IMPLICITE CONVERSION
var s="45";
var ic=s*1;
console.log("Number :"+ ic);
//EXPLICIT CONVERSION
var s1="72";
var ec = Number(s1);
console.log("Number : "+ec);
```

OUTPUT:

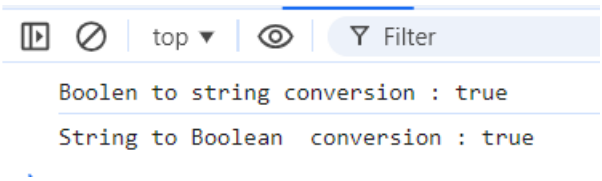


Task 27: Convert a boolean to a string and vice versa.

CODE:

```
var isStudent=String(true);
var s=Boolean("deeksha");
console.log("Boolen to string conversion : "+ isStudent);
console.log("String to Boolean  conversion : "+ s);
```

OUTPUT:



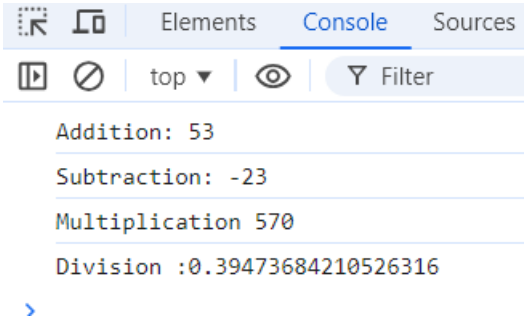
Task 28: Practice basic arithmetic operators (+, -, *, /, %).

CODE:

```
let a=15;
let b=38;
let add=a+b;
let sub=a-b;
let mul=a*b;
let div=a/b;
```

```
console.log("Addition: "+ add);
console.log("Subtraction: "+sub);
console.log("Multiplication "+ mul);
console.log("Division :"+div);
```

OUTPUT:

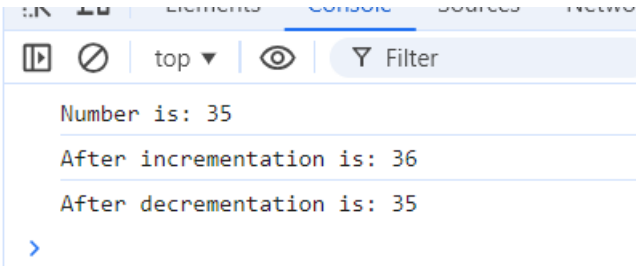


Task 29: Use the ++ and -- operators on a numeric variable.

CODE:

```
var num1=35;
console.log("Number is: "+num1);
num1+=1;
console.log("After incrementation is: "+num1);
num1-=1;
console.log("After decrementation is: "+num1);
```

OUTPUT:

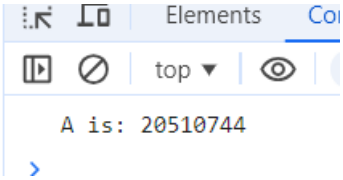


Task 30: Explore the precedence of operators by combining multiple operators in a single expression.

CODE:

```
 a=8*6^3+(145/5)**5-(98*4);
console.log("A is: "+a);
```

OUTPUT:



4: Comparisons, Conditional branching: if, '?'

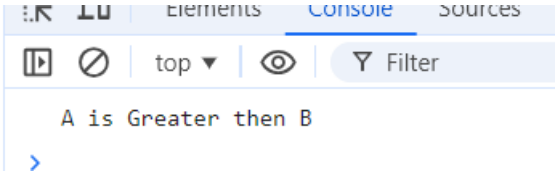
1. Comparisons:

Task 31: Compare two numbers using relational operators (>, <, >=, <=).

CODE:

```
let a=prompt("Enter the Value of A: ",2);
let b=prompt("Enter the Value of B: ",2);
if(a>b){
  console.log("A is Greater then B");
}
else if(a<b){
  console.log("A is Lesser then B");
}
else if(b>a){
  console.log("B is Greater then A");
}
else if(b<a){
  console.log("B is Lesser then A");
}
else{
  console.log("Both are Same!");}
```

OUTPUT:

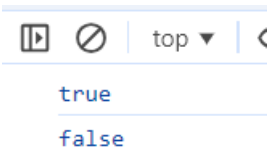


Task 32: Use equality () and strict equality (=) operators to compare different data types and note the differences.

CODE:

```
console.log(null==undefined);
console.log(null===undefined);
```

OUTPUT:

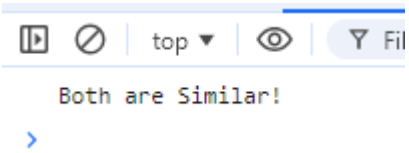


Task 33: Compare two strings lexicographically.

CODE:

```
let str1=prompt("Enter the String 1:",1);
let str2=prompt("Enter the String 2:",1);
if(str1==str2){
  console.log("Both are Similar!");
}
else if(str1>str2){
  console.log("string 1 is greater!");
}
else if(str2 > str1){
  console.log("string 2 is greater!");
}
```

OUTPUT:

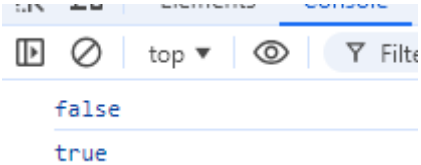


Task 34: Use the inequality (!=) and strict inequality (!==) operators to compare values.

CODE:

```
let a=50;
 let b='50';
 console.log(a!=b);
 console.log(a!==b);
```

OUTPUT:



Task 35: Compare null and undefined using both == and ===.

CODE:

```
console.log(null==undefined);
console.log(null===undefined);
```

OUTPUT:

```
true
false
```

2. Conditional branching: if, '?':

Task 36: Write an if statement that checks if a number is even or odd.

CODE:

```
let a =prompt("Enter the Value of A: ",1);
if(a%2==0){
  console.log("The Number is Even "+a);
}
else{
  console.log("The Number is Odd "+a);
}
```

OUTPUT:

```
The Number is Even 14
>
```

Task 37: Use nested if statements to classify a number as negative, positive, zero.

CODE:

```
var a=53;
 if(a!=0){
  if(a>0){
    console.log("positive");
  }
  else{
    console.log("Negative");
  }

 }
 else{
  ("Zero");
 }
```

OUTPUT:

```
positive
>
```

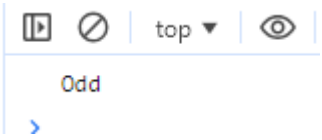
Task 38: Use the conditional (ternary) operator '?' to rewrite a simple if...else

statement.

CODE:

```
let n=39;
let a=(n%2==0)?"Even":"Odd";
console.log(a);
```

OUTPUT:

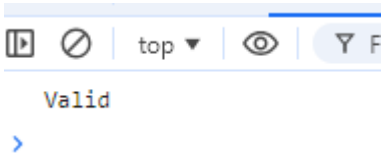


Task 39: Check the validity of a variable using the ? operator.

CODE:

```
et n=39;
let a=(n!=null && n!=undefined)?"Valid":"Not Valid";
console.log(a);
```

OUTPUT:

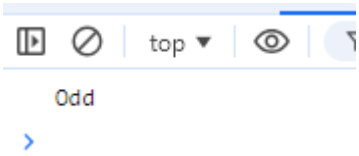


Task 40: Use the conditional operator to assign a value to a variable based on a condition.

CODE:

```
number = 7
result =(number % 2 == 0 )?"Even":"Odd";
console.log(result);
```

OUTPUT:



5: Logical operators, Functions

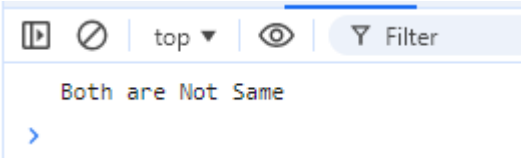
1. Logical operators:

Task 41: Evaluate various combinations of logical operators (&&, ||, !).

CODE:

```
let a=10;
let  b=20;
if(a!=b){
  console.log("Both are Not Same");
}
else if(a>0 && b>0){
  console.log("Both are Positive");
}
else if(a>=20 || b>20){
  console.log("Under Limit");
}
```

OUTPUT:



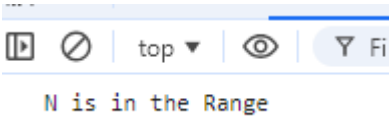
Task 42: Use logical operators to write a condition that checks if a number is in a

given range.

CODE:

```
n=prompt("Enter the value of n",1);
if(n<=100){
  console.log("N is in the Range");
}
else{
  console.log("N is not in the range");
}
```

OUTPUT:



Task 43: Use the NOT (!) operator to invert a boolean value.

CODE:

```
var b=true;
console.log(!b);
```

OUTPUT:

```
top ▾
false
›
```

Task 44: Evaluate the short-circuiting nature of logical operators.

CODE:

```
a=true;
b=false;
a1=a && b
a3=a || b
console.log(a1)
console.log(a3)
```

OUTPUT:

```
top ▾
false
true
```

Task 45: Compare two non-boolean values using logical operators and observe the result.

CODE:

```
let n= "Hello";
let n1 = 0;
let res1 = n && n1;
console.log(res1);
let a = "";
let a1 = "World";
let res2 = a || a1;
console.log(res2);
```

OUTPUT:

```
top ▾
0
World
```

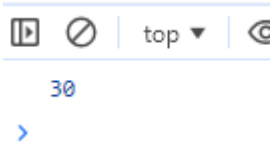
2. Functions:

Task 46: Write a function that takes two numbers as arguments and returns their sum.

CODE:

```
function addNo(n1,n2){
  return n1+n2;
}
let m=addNo(10,20);
console.log(m);
```

OUTPUT:

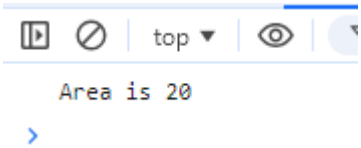


Task 47: Create a function that calculates the area of a rectangle.

CODE:

```
function calculatearea(l,b){
  return l*b;
}
let area = calculatearea(4,5);
console.log("Area is "+area);
```

OUTPUT:



Task 48: Declare a function without parameters and call it.

CODE:

```
function nopara(){
  console.log("Welcome to KCE");
}
nopara();
```

OUTPUT:

```
Welcome to KCE
>
```

Task 49: Write a function that returns nothing and observe the default return value.

CODE:

```
function demo(){

}
let res=demo();
console.log(res);
```

OUTPUT:

```
undefined
```

Task 50: Declare a function with default parameters and call it with different arguments.

CODE:

```
function calculatearea(l=16,b=30){
  return l*b;
}
let res= calculatearea(4,7);
console.log("Area= "+res);
```

OUTPUT:

```
Area= 28
>
```

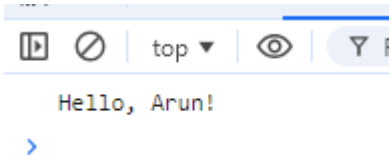
3. Arrow Functions:

Task 51: Declare a simple arrow function named greet that takes one parameter name and returns the string “Hello, name!”. Test your function with various names.

CODE:

```
const greet = (name) => `Hello, ${name}!`;
console.log(greet("Arun"));
```

OUTPUT:

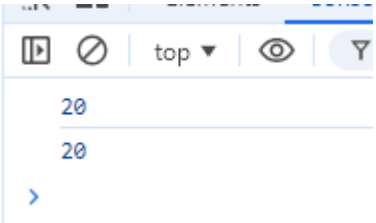


Task 52: Write an arrow function named add that takes two parameters and returns their sum. Validate your function with several pairs of numbers.

CODE:

```
const add = (a, b) => a + b;
console.log(add(7,13));
console.log(add(15, 8));
```

OUTPUT:

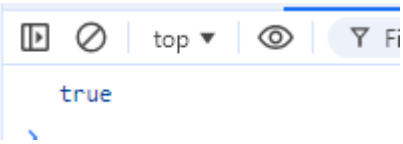


Task 53: Declare an arrow function named isEven that checks if a number is even.If the number is even, it should return true; otherwise, false. Remember that if the arrow function body has a single statement, you can omit the curly braces.

CODE:

```
const isEven = (number) => number % 2 === 0;
console.log(isEven(4));
```

OUTPUT:



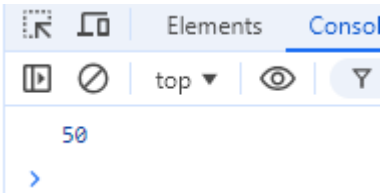
Task 54: Implement an arrow function named maxValue that takes two numbers as parameters and returns the larger number. Here, you'll need to use curly braces for the function body and the return statement.

CODE:

```
const maxValue = (a, b) => {
    return a > b ? a : b;
```

```
};
console.log(maxValue(50, 15));
```

OUTPUT:



Task 55: Examine the behavior of the this keyword inside an arrow function vs a traditional function. Create an object named myObject with a property value set to 10 and two methods: multiplyTraditional using a traditional function and multiply Arrow using an arrow function. Both methods should attempt tomultiply the value property by a number passed as a parameter. Check the value of this inside both methods.

CODE:

```
const myObject = {
    value: 10,
    multiplyTraditional: function(number) {
        console.log("Traditional Function:");
        console.log("this:", this);
        return this.value * number;
    },
    multiplyArrow: (number) => {
        console.log("Arrow Function:");
        console.log("this:", this);
        return this.value * number;
    }
};
console.log(myObject.multiplyTraditional(5));
console.log(myObject.multiplyArrow(5));
```

OUTPUT: