JavaScript Language - An Introduction to JavaScript, Code structure

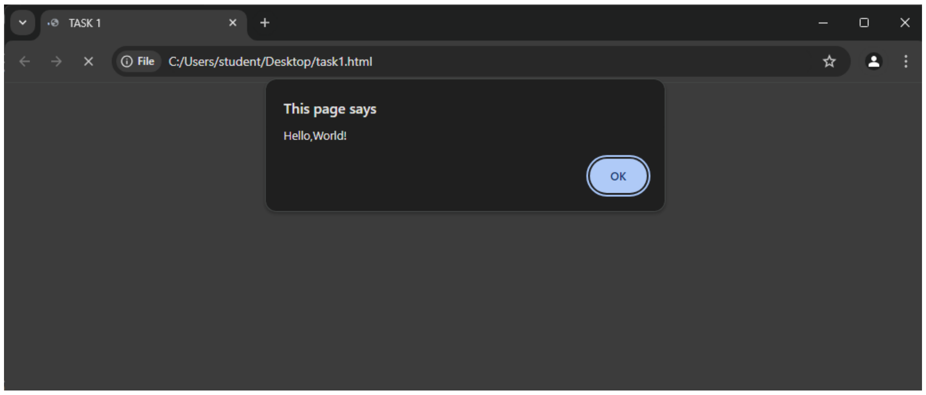
1.An Introduction to JavaScript:

<u>Task 1:</u> Write a simple script that displays "Hello, World!" on the web page using an alert box.

CODE:

```
<html>
    <head>
        <title>
            TASK 1
        </title>
    </head>
    <body>
        <script>
            alert("Hello,World!");
        </script>
    </body>
</html>
```

OUTPUT:



This page says

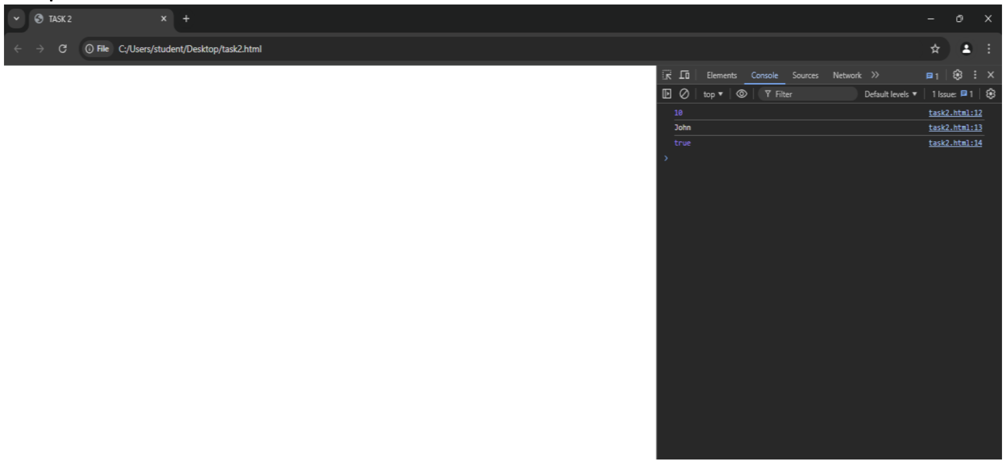
Hello,World!

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

Code:

```
<html>
    <head>
        <title>
            TASK 2
        </title>
    </head>
    <body>
        <script>
            let age=10;
            let name='John';
            let isboy=true;
            console.log(age);
            console.log(name);
            console.log(isboy);
        </script>
    </body>
</html>
```

Output:



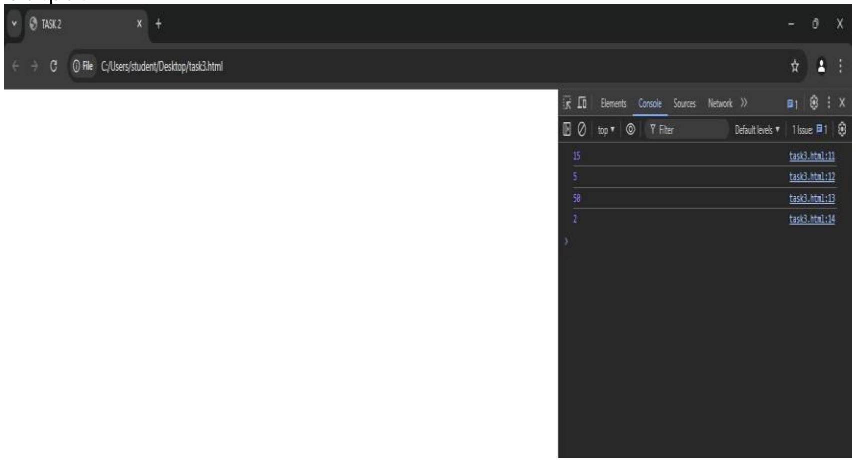
```
10                                    task2.html:12
John                                  task2.html:13
true                                  task2.html:14
```

<u>Task 3:</u> Use the console to perform basic math operations like addition, subtraction, multiplication, and division

Code:

```
<html>
    <head>
        <title>
            TASK 3
        </title>
    </head>
    <body>
        <script>
            let a=10;
            let b=5;
            console.log(a+b);
            console.log(a-b);
            console.log(a*b);
            console.log(a/b);
        </script>
    </body>
</html>
```

Output:

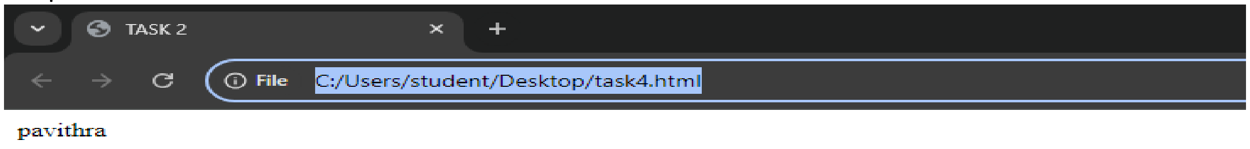


```
15                                    task3.html:11
5                                     task3.html:12
50                                    task3.html:13
2                                     task3.html:14
```

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

```
<html>
    <head>
        <title>
            TASK 4
        </title>
    </head>
    <body>
        <script>
            let a="pavi";
            let b="thra";
            document.writeln(a+b);
        </script>
    </body>
</html>
```

Output:

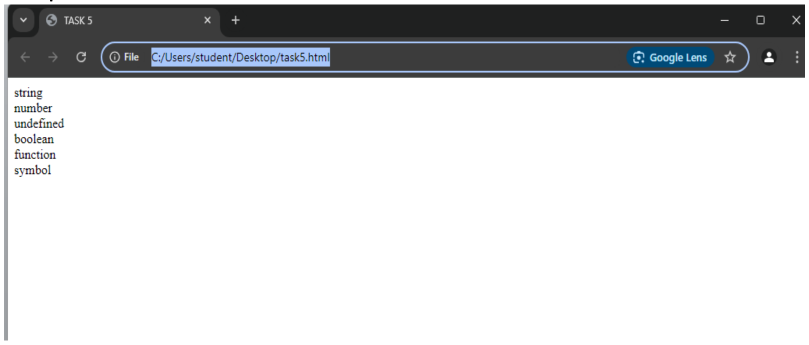


```
pavithra
```

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

```
<html>
    <head>
        <title>
            TASK 5
        </title>
    </head>
    <body>
        <script>
            let a="pavi";
            let  b=20;
            let c;
            let iscorrect=true;
            document.writeln(typeof(a)+"<br>");
            document.writeln(typeof(b)+"<br>");
            document.writeln(typeof(c)+"<br>");
            document.writeln(typeof(iscorrect)+"<br>");
            document.writeln(typeof(alert)+"<br>");
            document.writeln(typeof(Symbol("id"))+"<br>");
        </script>
    </body>
</html>
```

Output:


TASK 5

C:/Users/student/Desktop/task5.html

```
string
number
undefined
boolean
function
symbol
```

---

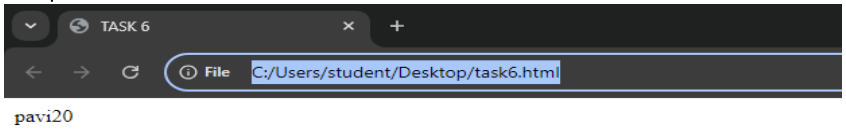
2. Code structure:

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

Code:

```
   <html>
<head>
<title>
TASK 6
</title>
</head>
<body>
<script>
let a="pavi";
let b=20;
/*
document.writeln(typeof(a)+"<br>");
 document.writeln(typeof(b)+"<br>");
 */
document.writeln(a+b);
//concatenate both string and number
</script>
</body>
</html>
```

Output:


TASK 6

C:/Users/student/Desktop/task6.html

```
pavi20
```

Diffference:

Syntax:

Single-line comments start with // and extend to the end of the line.

Multiline comments begin with /* and end with */, allowing for multiple lines of comments.

Usage:

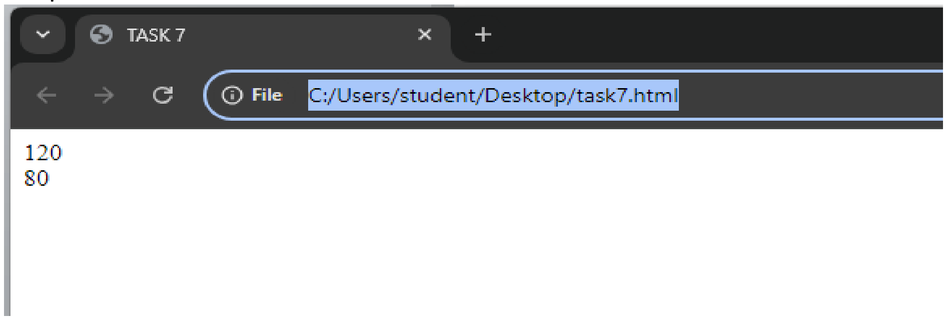
Single-line comments are typically used for brief, concise annotations.

Multiline comments are used for longer explanations or to comment out multiple lines of code at once.
Line Restriction:
Single-line comments are limited to a single line of text.
Multiline comments can span across multiple lines, allowing for more detailed or longer descriptions.

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

```
<html><head>
<title>TASK 7</title>
</head>
<body><script>
let a=100;
let b=20;
document.writeln(a+b+"<br>")
document.writeln(a-b);
</script></body></html>
```

Output:



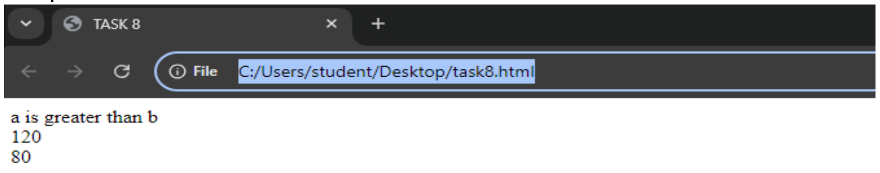
```
120
80
```

Diffference:
With Semicolons:
Clear and safe; each statement ends properly.
Without Semicolons:
JavaScript adds them automatically, but it can sometimes cause mistakes or errors.

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

```
<html><head>
<title>TASK 8</title>
</head>
<body><script>
let a=100;
let b=20;
if(a>b){
    document.write("a is greater than b"+"<br>");
}
else{
    if(a<b){
        document.write("a is smaller than b"+"<br>");
    }
    else{
        document.write("a is equal to b"+"<br>");
    }
}
document.writeln(a+b+"<br>")
document.writeln(a-b);
</script></body></html>
```

Output:



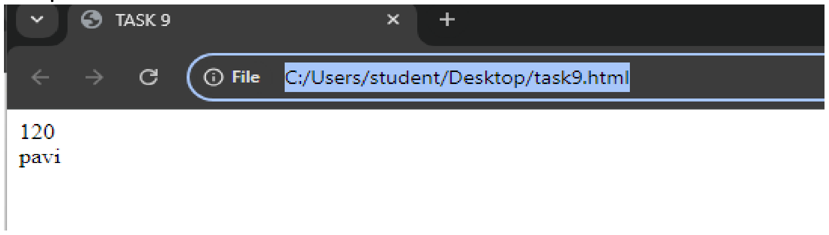
```
a is greater than b
120
80
```

Task 9: Declare multiple variables in a single line.

Code:

```
<html><head>
<title>TASK 9</title>
</head>
<body><script>
let a=100,b=20,name="pavi";
document.writeln(a+b+"<br>")
document.writeln(name);
</script></body></html>
```

Output:



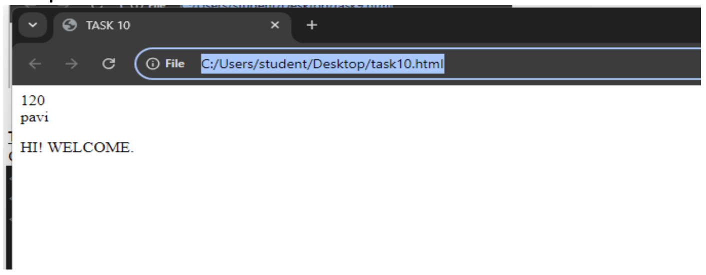
```
120
pavi
```

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

Code:

```
<html><head>
<title>TASK 10</title>
<script>
    let a=100,b=20,name="pavi";
    document.writeln(a+b+"<br>")
    document.writeln(name);
    </script>
</head>
<body>
    <p>HI! WELCOME.</p>
</body></html>
```

Output:

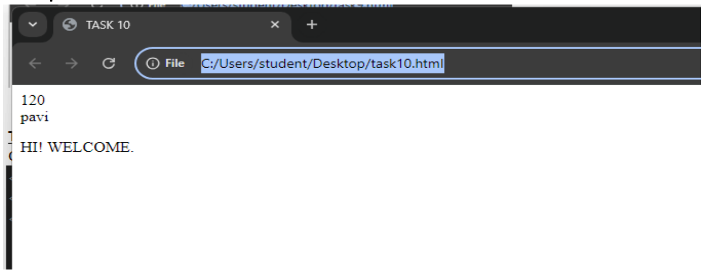


```
120
pavi

HI! WELCOME.
```

```html
<html><head>
<title>TASK 10</title>
</head>
<body>
   </p><script>
        let a=100,b=20,name="pavi";
        document.writeln(a+b+"<br>")
        document.writeln(name);
        </script>
        <p>HI! WELCOME.</p>
</body></html>
```

Output:



```
120
pavi

HI! WELCOME.
```

Diffference:

Script at the Top:

JavaScript executes before the HTML is fully loaded, which can cause issues if the script tries to access HTML elements that haven't been rendered yet.

script at the Bottom:

JavaScript executes after the HTML content is loaded, ensuring that all elements are available, and improving page load performance by allowing the HTML to render first.