What is JavaScript?

Javascript is a dynamic computer programming language. It is lightweight andmost commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

JavaScript was first known as LiveScript, but Netscape changed its name toJavaScript, possibly because of the excitement being generated by Java.

JavaScript made its first appearance in Netscape 2.0 in 1995 with thename LiveScript. The general-purpose core of the language has beenembedded in Netscape, Internet Explorer, and other web browsers.

The ECMA-262 Specification defined a standard version of the core JavaScriptlanguage.

- JavaScript is a lightweight, interpreted programming language.
- Designed for creating network-centric applications.
- Complementary to and integrated with Java.
- Complementary to and integrated with Java.
- Complementary to and integrated with HTML.
- Open and cross-platform

JavaScript is one of the popular programming languages that is used in most of the websites on the World Wide Web. It is also known as JS and is lightweight, cross-platform, and interpreted in nature.

Most of the time, JS is used to make webpages interactive and alive. For example, showing popups and changing page content when a button is clicked, validating users' emails on forms, and so on.

Basically, we can say that if you want to do anything related to websites and webpages, JavaScript is the language to go with.

Besides webpages, JavaScript is also used in game development, mobile app development, artwork creation, etc.
In short, it would be fair to conclude that javascript is the one language that rules the web.

History of JavaScript

JavaScript was created in 1995 by Brandan Eich, a Netscape programmer at that time. Initially, it was named Mocha and LiveScript. However, while taking the language public, it was named JavaScript.

The name was derived from the famous programming language Java to make it look more appealing.

Despite having a similar name, JavaScript has nothing to do with Java. It's an independent language with its own set of specifications. In recent times, JavaScript is also referred to as ECMAScript, and programs written in JavaScript are called scripts.

Client-Side JavaScript

Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to beinterpreted by the browser.

It means that a web page need not be a static HTML, but can include programsthat interact with the user, control the browser, and dynamically create HTML content.

The JavaScript client-side mechanism provides many advantages over traditionalCGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.

The JavaScript code is executed when the user submits the form, and only if allthe entries are valid, they would be submitted to the Web Server.

JavaScript can be used to trap user-initiated events such as button clicks, linknavigation, and other actions that the user initiates explicitly or implicitly.

Why Learn JavaScript?

You might be wondering why JavaScript, why not Python or Java. Well, the simple answer to this question is that JavaScript is used everywhere on the web.

If your goal is to dive into the field of web development, then you should choose JavaScript without any double thought. Here're some of the reasons why you should learn JavaScript.

- Beginner Friendly Language
- Versatile Language
- Best Language for Better Career
- High Paying Programming Language
- Huge Community

Why Learn JavaScript?

In the last article, we learned about the introduction of JavaScript programming language and we also introduced the language as the most popular programming language.

Apart from being the most popular language, there are various reasons why people should learn JavaScript

Beginner-Friendly Language

Unlike other programming languages, you can directly run JavaScript code on web browsers. You can directly jump into writing code without worrying about setting up the environment first.

This makes starting JS so much easier, especially for beginners who are learning JavaScript as their first programming language.

Also, JavaScript provides straightforward syntaxes that are easy to grasp for beginners.

JavaScript is a Versatile Language

JavaScript-based frameworks like Node, React, Angular, React-Native, and so on makes the language so versatile that you can build a complete application by just using JavaScript.

Not only that, you can build desktop, web, and mobile applications by only using JS.

Most Popular Programming Language

Yes, it is the world's most popular programming language and by a significant margin. There are many reasons that contribute to its popularity.

According to the latest survey by StackOverFlow, JavaScript is the most widely used programming language in the world for the tenth year in a row.

High Demand in Job Market

Being the most popular programming language, JavaScript developers are always in huge demand and it's increasing every month.

Not only that, JS developers are among the top paid professionals in the current job market.

Huge Community and Support

When working with JavaScript, even if you are creating small web applications, chances are that someone might have done the same, or something similar.

You can take help from different JavaScript communities of developers and enthusiasts. The community can offer small solutions to different approaches to doing the same thing.

Applications of JavaScript

Besides the reasons mentioned above, there are various applications of JavaScript due to which it has become the most desirable programming language which everyone should learn.

Web Applications

As mentioned earlier, JavaScript frameworks like React, Vue, Angular, and Node can be used to create full-fledged web applications.

In fact, popular applications like Microsoft Edge, Google Maps, and Uber are also built using JavaScript.

Web Development

From the early days, JavaScript was introduced to create web pages. It provides simple ways to add dynamic behaviors and special effects to the webpage.

For instance, the ability to load the content of a webpage without the need to reload the entire page, form validations, change page behaviors during button clicks, and so on.

With the help of JavaScript, we can even enable the user to load content into a document without them having to reload the entire page as well. Validations are one of the other important features that JavaScript provides us.

Many of the biggest tech companies make use of JavaScript to enhance the user experience even today.

Web Servers

With the introduction of Node.js, JavaScript developers now can create web servers from the ground up and take control of them.

If you don't know, servers are everywhere in today's web development world and allow you to transfer chunks of data without buffering.

JavaScript now being able to run outside the browser, via Node.js. It gives the developers the privilege to create web servers and take control of them. With Node.js, we can create servers from the ground up, and it's all JavaScript.

Game Development

Yes, JavaScript can even create Games, and it's this versatile nature of JavaScript that makes it so much

popular.

JavaScript can be combined with HTML5, and they together play a very important role in game development using JavaScript. There are different JavaScript libraries present that can be used to develop Games in JavaScript. Some of the famous games built using JavaScript are OpenHog, CrossCode, Tower Building, etc.

Mobile Applications

We can even use JavaScript to create mobile applications, and one of the most popular frameworks to do the same is React Native, which allows you to build cross-platform mobile applications.

Art

Don't be surprised seeing this point here as JavaScript does help in creating arts via Canvas elements. Creators can even draw and create 2D and 3D graphics by using the graphics on a web page.

This is one of those applications that not everyone knows about, but JavaScript excels when it comes to creating arts via Canvas elements, as by making use of graphics on a

web page, the creators can draw and create 2D and 3D graphics.

Hello World in JavaScript

A Hello World Program in JavaScript prints the text Hello World to the console.

For this we will use a HTML file and inside the HTML file, we will add a script tag. We will then write our JavaScript code inside that tag.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hello World - JavaScript Example</title>
</head>
```

```
<body>
    <script>
        console.log("Hello World");
    </script>
</body>
</html>
```

Explanation - We created a simple HTML5 document. Just focus on the <script> tag inside the body.

There you can find console.log("Hello World"); which prints the output Hello World onto the terminal.

To see the output,

run this index.html file in your browser

open the developer tool

go to the console where you will see "Hello World"

Limitations of JavaScript

We cannot treat JavaScript as a full-fledged programming language. It lacks thefollowing important features:

Client-side JavaScript does not allow the reading or writing of files. Thishas been kept for security reason.

JavaScript cannot be used for networking applications because there is nosuch support available.

JavaScript doesn't have any multithreading or multiprocessor capabilities. Once again, JavaScript is a lightweight, interpreted programming language thatallows you to build interactivity into otherwise static HTML pages.

Syntax

JavaScript can be implemented using JavaScript statements that are placedwithin the <script>... </script> HTML tags in a web page.

You can place the <script> tags, containing your JavaScript, anywhere within you web page, but it is normally recommended that you should keep it withinthe <head> tags.

The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear asfollows.

```
<script ...>

JavaScript code

</script>
```

The script tag takes two important attributes:

Language: This attribute specifies what scripting language you are using.Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.

Type: This attribute is what is now recommended to indicate the scriptinglanguage in use and its value should be set to "text/javascript".

So your JavaScript syntax will look as follows.

```
<script language="javascript" type="text/javascript">
JavaScript code
</script>
```

This function can be used to write text, HTML, or both. Take a look at thefollowing code.

```
<html>
<body>
<script language="javascript" type="text/javascript">
document.write ("Hello World!")
</script>
</body>
```

Whitespace and Line Breaks

JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs. You can use spaces, tabs, and newlines freely in your program andyou are free to format and indent your programs in a neat and consistent waythat makes the code easy to read and understand.

Semicolons are Optional

Simple statements in JavaScript are generally followed by a semicolon character,just as they are in C, C++, and Java. JavaScript, however, allows you to omit this semicolon if each of your statements are placed on a separate line. For example, the following code could be written without semicolons.

```
<script language="javascript" type="text/javascript">
<!--
var1 = 10
var2 = 20
//-->
</script>
```

But when formatted in a single line as follows, you must use semicolons:

```
<script language="javascript" type="text/javascript">
<!--
var1 = 10; var2 = 20;
//-->
</script>
```

Note: It is a good programming practice to use semicolons.

Case Sensitivity

JavaScript is a case-sensitive language. This means that the language keywords,variables, function names, and any other identifiers must always be typed with aconsistent capitalization of letters.

So the identifiers Time and TIME will convey different meanings in JavaScript.NOTE: Care should be taken while writing variable and function names in JavaScript.

Why use comments?

Comments are valuable components and part of any programming language, including JavaScript. Below are the two most common uses of comments in JavaScript

1. Increasing Readability

Comments help in improving the readability of the code

Programmers can refer to them if they visit the code after some time.

New programmers can use comments to understand the code before starting to work.

2. Debugging

Comments are non-executable statements, so if we know that a particular part of the code is causing an error, we can easily comment on that part while debugging.

Types of Comments

In JavaScript, there are two types of comments. These are

1. Single Line Comment
2. Multi-Line Comment

Single Line Comment

The single-line comment starts with the // symbol. Any text that is present after // up to the end of the line will be ignored by JavaScript.

Consider the code that is shown below in which we are creating two single-line comments.

index.html

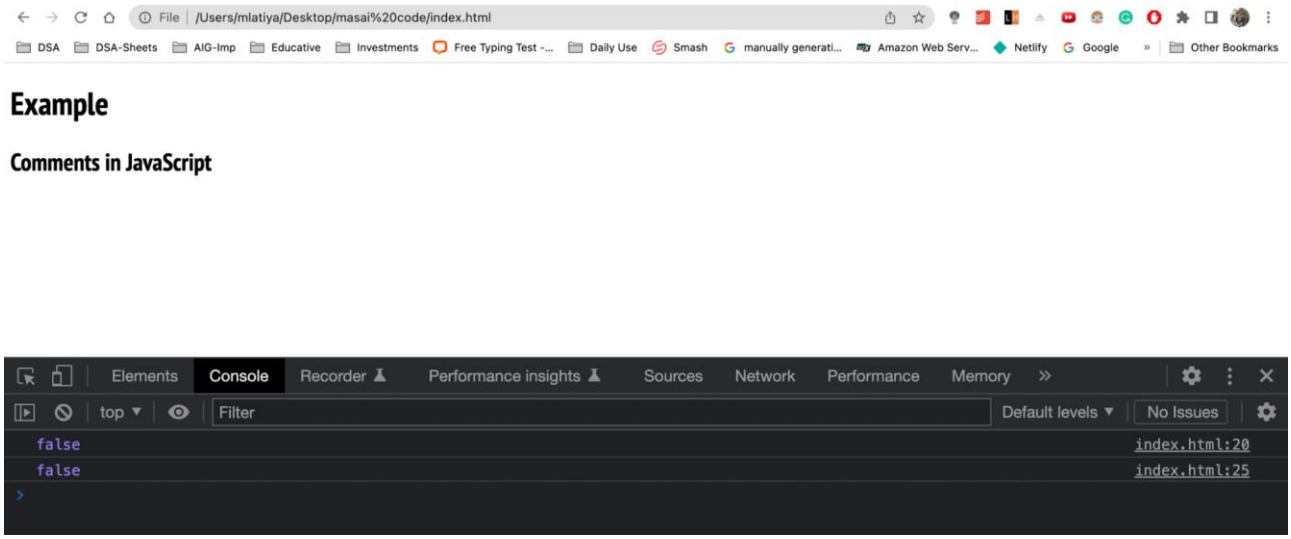
```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible"
content="IE=edge">
        <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
        <title>Comments in
JavaScript</title>
</head>
<body>
        <h1>Example</h1>
        <h2>Comments in JavaScript</h2>
      <script>
        let str = 0;
```

```
        // Using the Boolean method
        console.log(Boolean(str));
        str = "";
        // Using the Boolean method
        console.log(Boolean(str));
    </script>
</body>
</html>
```

Explanation - In the above example, we have created two comments // Using the Boolean method just before the console.log() function.

We will get the following output if we run the above code in the browser.

Output:



Multi-Line Comment

The multi-line comment starts with the /* symbol and ends with the */ symbol. Any text that is present between these /* and */ symbols will be ignored by JavaScript.

Consider the code that is shown below in which we are creating multi-line comments.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible"
content="IE=edge">
```

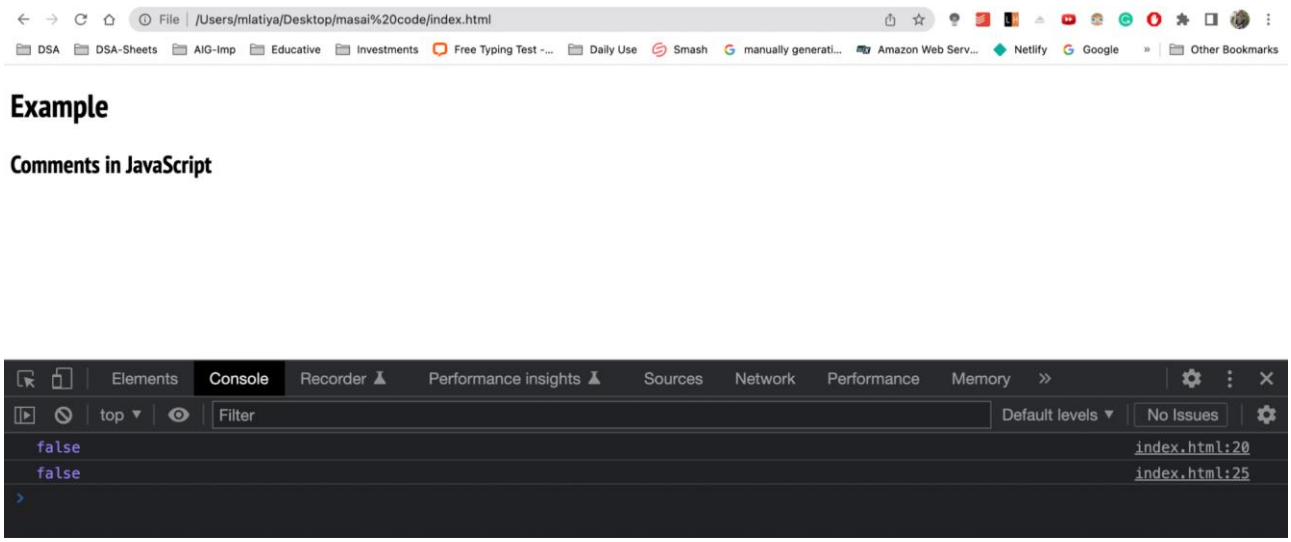
```
        <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
        <title>Comments in
JavaScript</title>
</head>
<body>
        <h1>Example</h1>

        <h2>Comments in JavaScript</h2>
        <script>
        let str = 0;
        /*
            Using Boolean method
            in the console.log()
            shown below
        */
        console.log(Boolean(str));
        str = "";
        /*
            Using the Boolean method
```

```
                in the console.log()
                shown below
            */
            console.log(Boolean(str));
    </script>
</body>
</html>
```

Explanation - In the above example, texts in between /* and */ (Using the Boolean method in the console.log() shown below) are two multi-line comments in JavaScript.

If we run the above code in the browser, we will get the following output.

Output:



Comments to prevent the execution of the code

We can even prevent some particular part of the code from being executed if we comment on that part of the code.

Consider the code shown below.

index.html

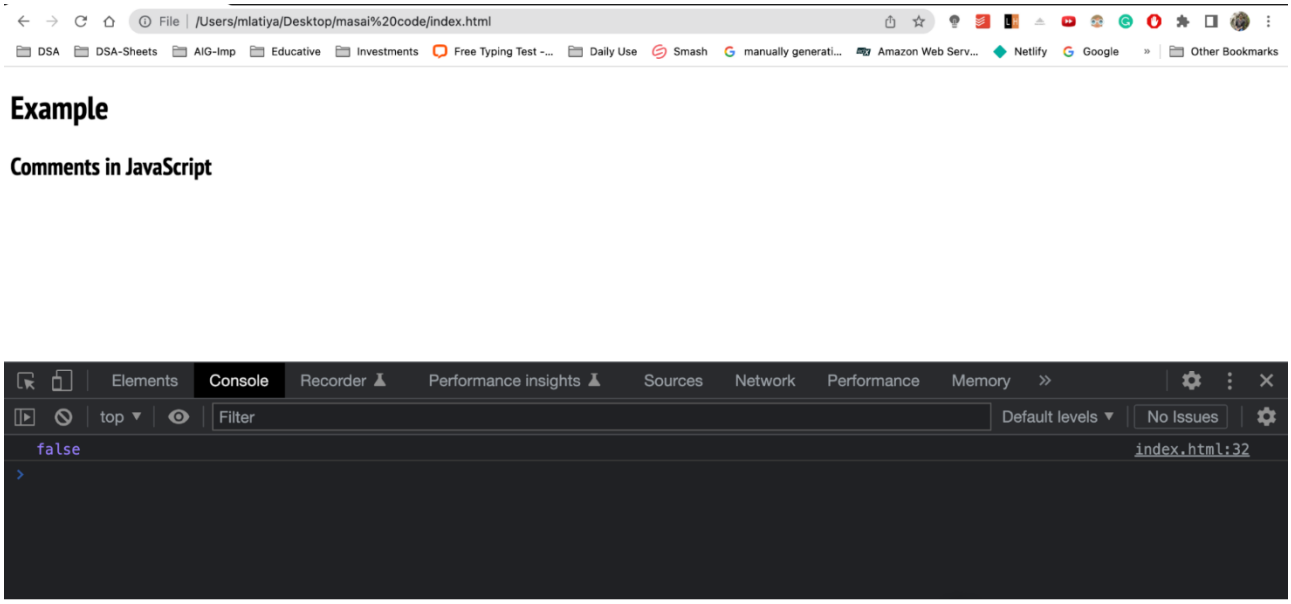
```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible"
content="IE=edge">
        <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
        <title>Comments in
JavaScript</title>
</head>
<body>
        <h1>Example</h1>
        <h2>Comments in JavaScript</h2>
```

```
        <script>
        let str = 0;
        /*
            commenting the console.log()
            functions call below
        */
        // console.log(Boolean(str));
        str = "";
        /*
            Using Boolean method
            in the console.log()
            shown below
        */
        console.log(Boolean(str));
    </script>
</body>
</html>
```

Explanation - In the above code, we commented out a console.log(Boolean(str)); using a single line comment. When we run the code shown above, the interpreter will

ignore the commented part.

Output:



Commenting out code is important, especially when we are debugging code. Because instead of removing code, we can comment them out, so they won't be executed and then uncomment whenever needed.

Good comments vs. Bad Comments

While comments for sure help us to increase readability and code experience, not all comments are good for the code.

Below are some of the ways with which you can distinguish good and bad comments:

Bad Comments:

- Too much explanation
- Misleading data
- Not intention-revealing
- Redundant comments

Good Comments:

- Intent Explanation
- Informative
- Warning of Consequences

PLACEMENT OF SCRIPT TAG

There is a flexibility given to include JavaScript code anywhere in an HTML document. However the most preferred ways to include JavaScript in an HTMLfile are as follows:

Script in <head>...</head> section.

Script in <body>...</body> section.

Script in <body>...</body> and <head>...</head> sections.

Script in an external file and then include in <head>...</head> section.

In the following section, we will see how we can place JavaScript in an HTML file

in different ways.

JavaScript in <head>...</head> Section

If you want to have a script run on some event, such as when a user clickssomewhere, then you will place that script in the head as follows.

```
<html>
<head>
<script type="text/javascript">
<!--
function sayHello() {alert("Hello World")
}
//-->
</script>
</head>
<body>
```

JavaScript in <body>...</body> Section

If you need a script to run as the page loads so that the script generates contentin the page, then the script goes in the <body> portion of the document. In thiscase, you

would not have any function defined using JavaScript. Take a look at the following code.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
<!--
document.write("Hello World")
//-->
</script>
<p>This is web page body </p>
</body>
</html>
```

JavaScript in <body> and <head> Sections

You can put your JavaScript code in <head> and <body> section altogether asfollows.

```
<html>
```

```
<head>
<script type="text/javascript">
function sayHello() {alert("Hello World")
}
//-->
</script>
</head>
<body>
<script type="text/javascript">
<!--
document.write("Hello World")
//-->
</script>
<input type="button" onclick="sayHello()" value="Say
Hello" />
</body>
</html>
```

JavaScript in External File

As you begin to work more extensively with JavaScript, you will be likely to findthat there are cases where you are reusing identical JavaScript code on multiplepages of a site.

You are not restricted to be maintaining identical code in multiple HTML files.The script tag provides a mechanism to allow you to store JavaScript in an external file and then include it into your HTML files.

Here is an example to show how you can include an external JavaScript file inyour HTML code using script tag and its src attribute.

```
<html>
<head>
<script type="text/javascript" src="filename.js" ></script>
</head>
<body>
.....
</body>
</html>
```

To use JavaScript from an external file source, you need to write all your JavaScript source code in a simple text file with the extension ".js" and theninclude that file as shown above.

For example, you can keep the following content in filename.js file and then you can use sayHello function in your HTML file after including the filename.jsfile.

```
function sayHello() {alert("Hello World")
}
```

Introduction to Printing Output

Generally, when we perform some actions on the website, like clicking a button or providing input to a text area, we get some response in return. Similarly, in programming as well, we want to get some response after running our code.

These responses are known as output. The output may be simple plain text, an image, or an entire document file.

In this article, we will explore all the different ways of printing output in JavaScript.

Different Ways to Print Output

When it comes to JavaScript, there are four different ways with which we can print output to the terminal or console.

- Using `console.log()`

- Using window.alert()
- Using document.write()
- Using innerHTML

console.log()

The console.log() method in JavaScript is used when we want to write a message to the console. Whatever message we want to print to the console, we pass it inside the log() method as an argument.

Let's create a simple example of the console.log() method.

Consider the index.html code shown below.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
        <title>console.log() - JavaScript Example</title>
</head>
<body>
        <p>
                Some text.
        </p>
        <script>
                console.log("Hello There!");
        </script>
</body>
</html>
```

Explanation - In the above code, we have a paragraph tag first and a script tag. Inside the script tag, we have used the console.log() method to print "Hello There!".

Once we run the code, we will see "Some text." printed on the screen, and when we open the browser's console, we will get Hello There!.

window.alert()

As the name suggests, the windows.alert() method prints the output data as an alert in the browser. Let's see an example.

Consider the index.html code shown below.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>window.alert() - JavaScript Example</title>
</head>
<body>
    <p>
        Some text.
    </p>
    <script>
```

```
        window.alert("Welcome to JavaScript")
    </script>
</body>
</html>
```

Explanation - In the above code, we are making use of the window.alert() method to print the text "Welcome to JavaScript" in the browser window.

One important point to note here is that we can even avoid the window keyword. It's because the window is a global object in JavaScript, so it's allowed to access its method and properties directly.

Consider the index.html code shown below.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>window.alert() - JavaScript Example</title>
</head>
<body>
    <p>
        Some text.
    </p>
    <script>
        alert("Welcome to JavaScript ")
    </script>
</body>
</html>
```

Explanation - As you can see, we have removed the window keyword and when we run this code, we will get the same output as before.

innerHTML()

innerHTML in JavaScript is used to add required texts to the specific HTML element and print it.

Let's say that we have a very simple div, in which we are simply printing a text to the browser screen, and we want to replace the text with the text of our choice. In this case, we can make use of the innerHTML property.

Consider the index.html code shown below.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>innerHTML - JavaScript Example</title>
</head>
<body>
    <div id="simpleDiv">
```

```
        This is a simple Div.
    </div>
    <script>

document.getElementById("simpleDiv").innerHTML =
"No, it is more than just a simple div"

    </script>
</body>
</html>
```

Explanation - In the above code, inside the body tag, we have a div with id simpleDiv, which is printing the text "This is a simple Div.".

Now, instead of this text, suppose we want to print the text of our own. Then, we can make use of the innerHTML. First, we will call the getElementById() method to select the HTML element where we want to make the changes and then assign a new value to the innerHTML property.

```
document.getElementByID("simpleDiv").innerHTML = "No,
it is more than just a simple div"
```

We can even place a numeric value as well in the innerHTML property.

Consider the index.html code shown below.

index.html

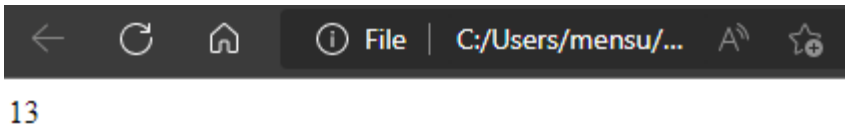
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>innerHTML - JavaScript Example</title>
</head>
<body>
    <div id="simpleDiv">
        This is a simple Div.
    </div>
    <script>
```

```
document.getElementById("simpleDiv").innerHTML = 13
    </script>
</body>
</html>
```

Explanation - In the above code, I used a numeric expression to be placed as the value of the innerHTML property.

Once we run the above code in the browser, we would see the following text printed.

Output:



13

document.write()

The document.write() method prints the output text on the browser's webpage. Let's see an example.

Consider the index.html code shown below.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>document.write() - JavaScript Example</title>
</head>
<body>
    <script>
        document.write("Shrikant.");
    </script>
</body>
</html>
```

Explanation - In the above code, we are using the document.write() method to print the text "Masai School is Awesome.".

Note: If we use document.write() after the webpage has finished loading, it will overwrite everything that is on the webpage. Let's see one example.

Consider the index.html code shown below.

index.html

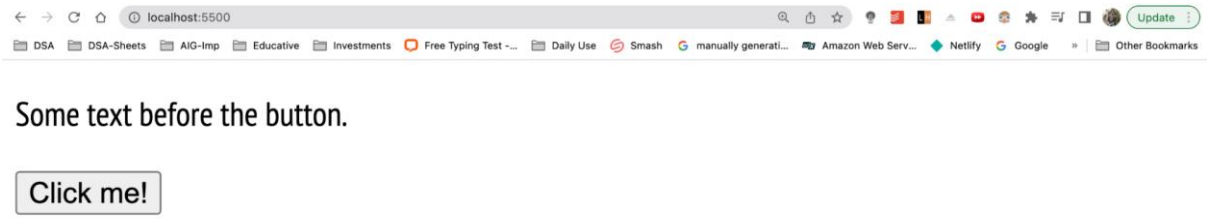
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>document.write() - JavaScript Example</title>
</head>
<body>
    <p>
        Some text before the button.
    </p>
```

```
    <button onclick="document.write('New text overwrites the existing text');" >
        Click me!
    </button>
</body>
</html>
```

Explanation - In the above code, there's a p tag and a button tag. Inside the button tag, we have used the onclick method that is calling document.write. Because of this, the document.write() method is only called when we click on the button.

Once we run the code, we will see the text inside the paragraph tag and a button.

Output



Here, this page has finished loading. Now, if we click the

button, the document.write() method is called, which will overwrite the existing page content with a new one.

Output

You can see that only "New text overwrites the existing text" is shown now. Because of this behavior, we should never use document.write() after the page has finished loading.

Working of document.write()

In the above code, the p tag is printing some text and the button has an onclick event set up. Here, when we click the button, the document.write() method is called.

Initially, when we run the code, the web page is completely loaded, which will print the text and show the button. At this point, the onclick event is not triggered. Now, when we click the button, the document.write() method is called, which will overwrite everything on the webpage and print the text inside it.

While the above four methods are the ones that are used when we want to print some text or value to the terminal or the console in JavaScript, there's one more method that might give the impression that it is used to print a text to the browser console or your terminal. This method is called the window.print() method.

window.print()

The `window.print()` method is different from all the methods that are mentioned above as it is used when we want the browser to print the content that we have on the current browser window.
JavaScript ,by default, doesn't have any print method.
Consider the index.html code shown below.

index.html

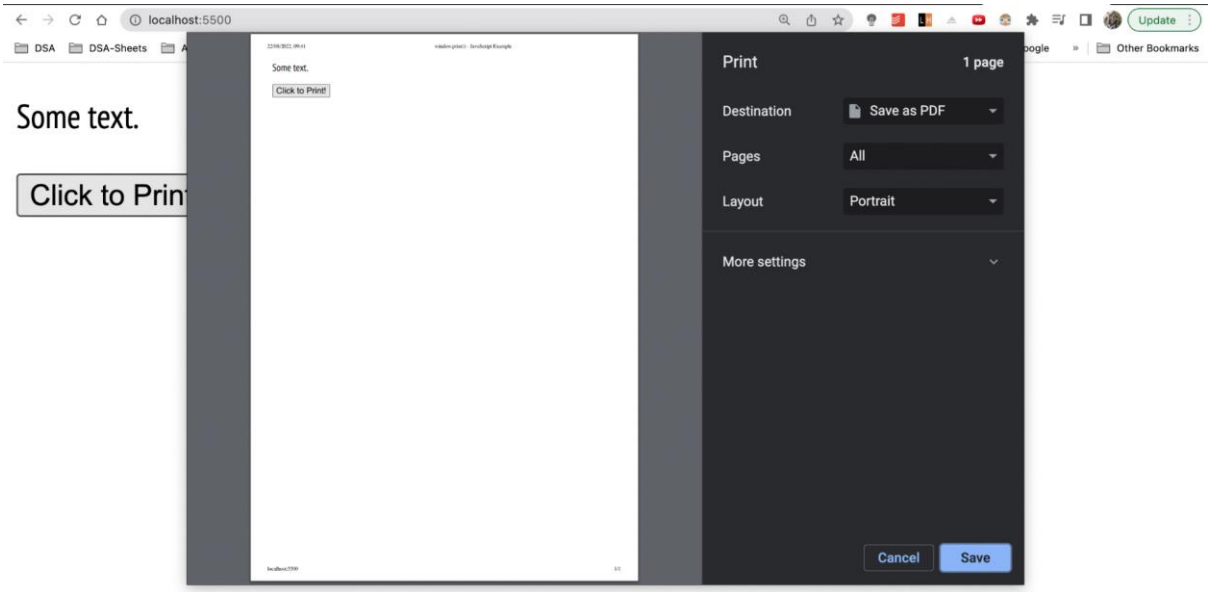
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible"
content="IE=edge">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>window.print() - JavaScript Example</title>
</head>
<body>
    <p>
        Some text.
    </p>
    <button onclick="window.print();">
        Click to Print!
    </button>
</body>
</html>
```

Explanation - In the above code, we have a paragraph tag first, and then inside the script tag, we have created a button with an onclick attribute that calls the window.print().

Once we run the above code in the browser, we would see the following text printed in the browser's console.

Output:



After clicking on the button, you will be prompted with a printing screen that will ask you whether you want to print the content of the current window or not.