

INTERACTIVITY WITH JAVASCRIPT COURSE NOTES

Contents

[Introduction to JavaScript 1](#_Toc113717341)

[Basic Terminologies in Computer Science 1](#_Toc113717342)

[The Document Object Model (DOM) 2](#_Toc113717343)

[How to write JavaScript Code? 5](#_Toc113717344)

[Variables in JavaScript 10](#_Toc113717345)

[Data Types in JavaScript 13](#_Toc113717346)

[Operators and Expressions in JavaScript 14](#_Toc113717347)

**INTERACTIVITY WITH JAVASCRIPT**

After learning HTML and CSS comprehensively, it is time to learn one of the world’s widely popular and extensively used web language – JavaScript (JS).

JavaScript, as the name suggests, is a **scripting** language that is used to create dynamic content on the webpage, with which the user can interact.

A **Scripting language** or **Script language** is a programming language that is used to manipulate, customize, and automate the facilities of an existing system. Such languages are usually interpreted at runtime rather than compiled, unlike our general programming languages.

JavaScript is used for enhancing User Experience for your website, and a great way to improve your site’s accessibility. Please note that **JavaScript is not related to Java**, and both are completely different.

# Introduction to JavaScript

If you've done HTML and CSS, you haven't necessarily done what we call real programming, at least not what us computer scientists call real programming. Instead, let us talk about JavaScript as about really definite data manipulation, and in order to do that, there are a few things you need to learn how to do:

* learn what are variables and how to store them.
* learn how to set decision points,
* learn looping and writing functions,
* learn how you can get data from the browser (main use of JavaScript)

Every time you load a web page, you can actually get back the title of the page, the URL of your page, a lot of different information that you know is there, but you cannot access it. JavaScript helps us do exactly that.

The other thing we can do with JavaScript, which is really where it gets a little bit more exciting, is that we can manipulate the DOM that the browsers use to create webpages.

# Basic Terminologies in Computer Science

When we will use JavaScript to access data of a webpage, we will be using some basic tools and terminologies. Let us look at them.

**Variables**

It is important to store that data into some container, or what we call identifiers. These identifiers can be accessed later on, when we need them.

The names of these identifiers are called ***Variables*** in computer science. Variables are basically aliases or nicknames for a memory location, where the data is stored.

We will be using a lot of variables in JavaScript to access and use the data extracted from a webpage.

**Decision Points**

Decision Points are pretty similar to filtering a data based on a condition. In this way, we are controlling the kind of data we are receiving from the webpage. We do this by using control structures / statements.

These decision points help us in specifically working on the data, that we need, and not getting everything, which will make things complicated.

**Looping**

Looping is the reason a computer is much more efficient than humans. Looping is when you repeatedly execute a code block (multiple lines of code) as many times as we, the programmers, want. It is the main power of computers.

It avoids rewriting code that will be used again and again. One basic shortcoming with loops is that if you have to use 100 loops inside a program, then you have to copy paste its code 100 times. This makes your code organization really messy, for someone trying to understand it.

**Functions**

Functions are a definitive way of addressing the shortcoming in loops. It contains a block of code, represented by an identifier, followed by brackets, in which you can manipulate the code block using entries inside the brackets (also called parameters).

This way, you just have to write 100 statements, containing a single word, which will be better for people trying to understand your code, and for you, the programmer, since you won’t have to copy-paste the same code again and again.

# The Document Object Model (DOM)

Every webpage is built upon this DOM. And what it means is that our pages are structured like a tree. We have one parent and possibly a few children. And the page as you add more and more things it just gets deeper and deeper.

The DOM is very essential to JavaScript, since it is the way the web pages are rendered by the browsers, and also, the way that JavaScript goes inside a webpage to extract data from it.

For eg. We have a webpage with the following code –

<html>

<head> <title> --- </title> </head>

<body>

<div> --- </div>

<div id=3> --- </div>

<div> --- </div>

</body>

</html>

This webpage will be interpreted as a DOM structure model. It will have nodes containing elements of HTML, and each element will have attributes defined by CSS.

The content inside the webpage will be represented by the DOM, while scripting languages like JavaScript, use this DOM to interact with the document.

**How does JavaScript use DOM ?**

the JavaScript has something called an **API**, or an **Application Programming Interface**. An API is a bridge between two computer languages. In other words, an API is just a way for someone else to write code, and then you to use it, to interact with their programs.

No matter which browser you're using, Firefox, Chrome, different things like that. And no matter which scripting language you're using, in our case we're going to use JavaScript. The API is always the same. The way you interact is always gonna be the same. And this is really important, because it's nice to know that if you learn this you won't have to learn something different later on.

**DOM Objects**

The DOM Model consists of various objects inside it, that represent different parts of a webpage. Some of these objects are listed below –

* Document – The document is the root of the webpage. if you wanted to, you could find out what the document URL is, what the height of the document is, all the different links in the document, the document background color. If there's an attribute you've used to style your page, you can find it using JavaScript in the API.
* Element / Node - You can have the body, and inside the body, you could have divs, paragraphs, links, list items. Each one of those is a node in the tree. And you can find it using the API.
* NodeList – Sometimes, when you have many child tags inside a tag, then you will get a list of all the nodes (elements) from the API.
* Attributes – Elements have various attributes, as defined by HTML. To extract these attributes, the DOM model has allotted an object that will return the attributes of an element.

Now, we understand that the DOM structure consists of document, nodes / elements, nodelist and attributes. Now, let us see some specific APIs that will help us extracting some particulars from the webpage.

**Specific APIs**

Each of the APIs listed below can be used to extract some information of the webpage, as per our requirements. These are –

* document.getElementById(id)
* document.getElementsByClassName(class)
* element.innerHTML
* element.style
* element.setAttribute(attribute,value)
* element.removeAttribute(attribute)

These APIs are very useful in not just getting information, but also in modifying the code of the webpage using a scripting language like JavaScript.

Now, let us learn more about interactivity first, since our main target of using JavaScript is to manipulate the webpage to increase interactivity.

**Interactivity**

Previously, we have used some components of HTML and CSS to introduce dynamic elements in our webpage. Components like pseudoclasses, and pseudoelements, or the properties like hover, focus, transform etc were used to make the webpage interactive.

However, it is important to understand that **these elements are not really interactive**. These elements and properties can only go so far. These are, what we call, temporary changes. Let us ask a question.

**What can JavaScript do that HTML and CSS cannot?**

The main thing that JavaScript does is – it reads and writes HTML documents. In the case of HTML and CSS, we can only write documents, and they are read or what we call rendered, by applications having APIs. Each browser has multiple APIs that it uses to render a webpage.

JavaScript is much vaster than this. It is a proper programming language, that is used to manipulate web pages. It is not limited to just reading and writing documents. But, for now, we will focus on Output.

# How to write JavaScript Code?

It is important to understand that since JavaScript is a scripting language, it does not have a console of its own, where you can print things. JavaScript code is executed on runtime, and hence, it is written in HTML code only.

If you have just some simple codes of 5-6 lines, you can use the <script> tag inside the <head> tag, or any other tag and write your JavaScript code there. It follows the following syntax –

<html>

<head>

<script>

--- javascript code here ---

</script>

</head>

<html>

This can also be called as **Internal JavaScript**, similar to internal CSS. This is limited to the HTML code of a single webpage.

If you have long lines of JavaScript code, and / or you need to use it in multiple pages, in that case, we use the <script> tag along with the src attribute. It has the following syntax –

<script src=“path-to-the-file.js”></script>

This can also be called as **External JavaScript**, similar to external CSS. For creating the external script file, we need to save it with the extension \*.js. In this case, we need to close the <script> tag as well.

**Programs and Statements in JavaScript**

The external JavaScript file is called a JavaScript Program. Let us learn what a program is.

A **Computer Program** is a list of instructions to be executed by a computer. In a programming language, these programming instructions are called statements.

A **JavaScript program** is a list of programming statements written in JavaScript. **JavaScript Statements** are composed of Values, Operators, Expressions, Keywords, and Comments. Sometimes, they contain Output as well.

The following is the example of a JavaScript program –

var radius=14;

var pi=3.14;

var str="Area";

var bool=true;

document.write('The radius is: <font color="red">',radius,'</font><br>');

document.write('The value of pi is: <font color="red">',pi,'</font><br>');

document.write('The string is: <font color="red">',str,'</font><br>');

document.write('The correct value is: <font color="red">',bool,'</font>');

Each of the line in this JavaScript program is a Statement. Now, let us learn about each component of a statement individually.

**Output in JavaScript**

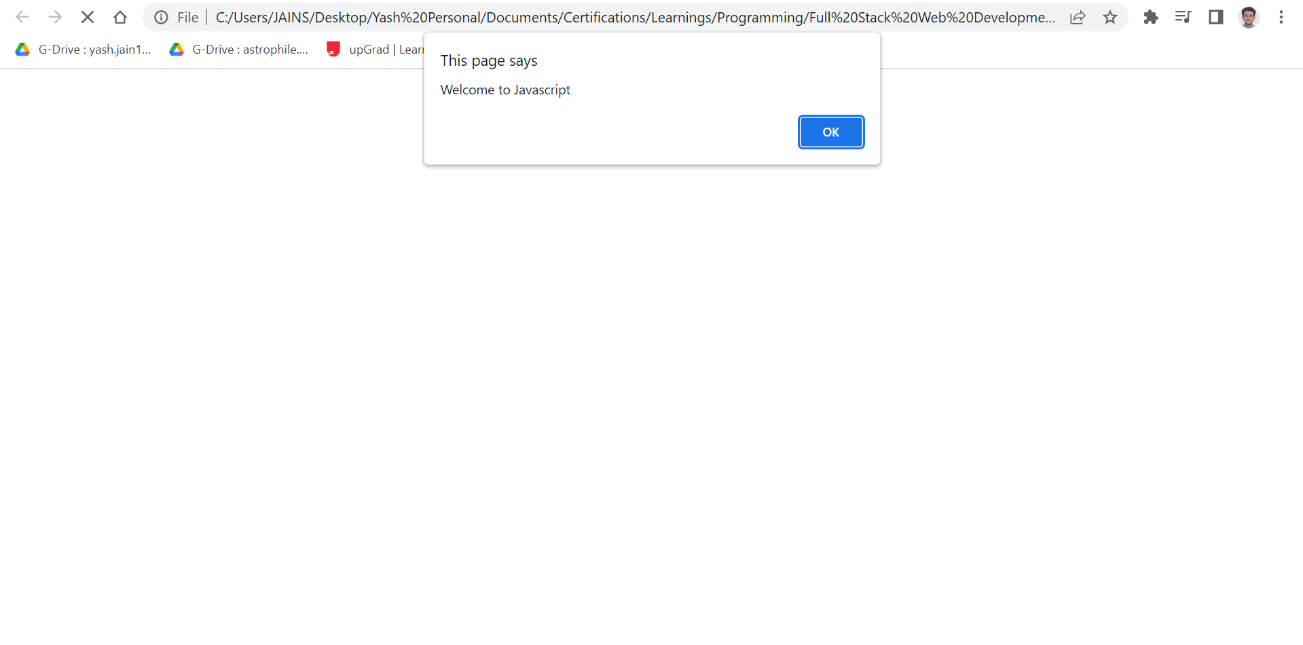
Whenever you learn a new programming language, one of the first things you want to do is find out how can I make things happen. How can I have things print out to the screen or generate some sort of output.

JavaScript doesn't have a built-in print function. So instead, you have four or five different ways where you can generate different things. Data is typically displayed via –

alert() – This is a pop-up window that displays information. It contains a string (collection of alphanumeric characters) inside double quotes inside the parenthesis, which will contain your output message. It follows the following syntax –

alert(“Your message here”);

Let us see how this would look on a browser. It would look like the following –

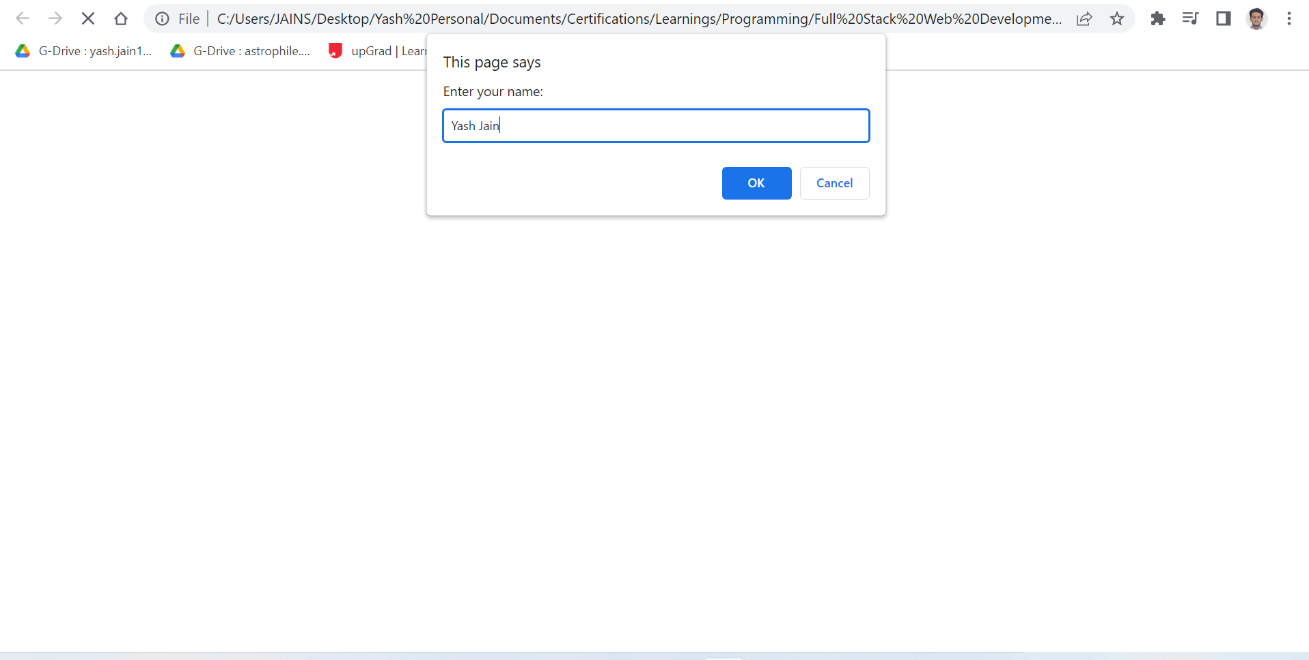


This would just be a popup message, and it won’t show anything on the webpage, after you click Ok. We have another function for popups.

prompt() – This function is similar to the alert() function, the only difference being that it takes input from the user. It follows the following syntax –

prompt(“Enter your name”);

This function will take the input from user in the popup window. Just like the alert() function, it won’t print anything on the webpage, after clicking Ok. Let us see it in action –



Both these functions are used for popups. However, to actually print something on the webpage, we use some other methods.

document.write() – This function is an example of JavaScript using the DOM. It is telling the browser to take the document object of the DOM (which will be extracted via the API automatically) and write the content inside the parenthesis to the document. Let us see its syntax –

document.write(“string”);

We can simply write this statement in an internal JS syntax using the <script> tag, and it would be displayed. We can also use the HTML tags inside the function. See the following example –

<!DOCTYPE html>

<html lang='en'>

<head>

<script> document.write('Welcome to JavaScript')</script>

</head>

<body><h1>This is how we use the <code> document.write()</code> function in JavaScript. </h1>

<script> document.write('We can also use <code>script</code> inside the <code><body></code> tag, like in this statement. We have also used HTML tags inside the function.'); </script>

<h5> All done ! </h5>

</body>

</html>

Let us see the output for the above code.

Graphical user interface, text, application, Word

Description automatically generated

This feature may not seem useful now, since we are only printing statements (which we could have done using the <body> tag itself), but further, when we use variables, then it will be used extensively.

We need to realize that it is probably not the best way to print output. Because sometimes if you're misusing it, you can overwrite other things that exist. So document.write() is something you just want to use when you're beginning, and you don't know some of the more complex methods.

element.innerHTML – This is one of the complex methods to write things using JavaScript. It uses an element, returned from API, and modifies the content inside it. It follows the following syntax –

element.innerHTML = “string here”;

It is clearly visible that we are not using a parenthesis here. This is because this is not a function; it is a method used by JavaScript to modify the contents of any element of a webpage.

The element will be extracted using different APIs, like we studied above – the getElementById(), or getElementByClassName().

Let us see an example.

<!DOCTYPE html>

<html lang=’en’>

<head>

<body>

<h4 id=“test”> This is Test </h4>

<h4 id=“test”> This is Test </h4>

<script>

document.getElementById(‘test’).innerHTML = “Hello World !”;

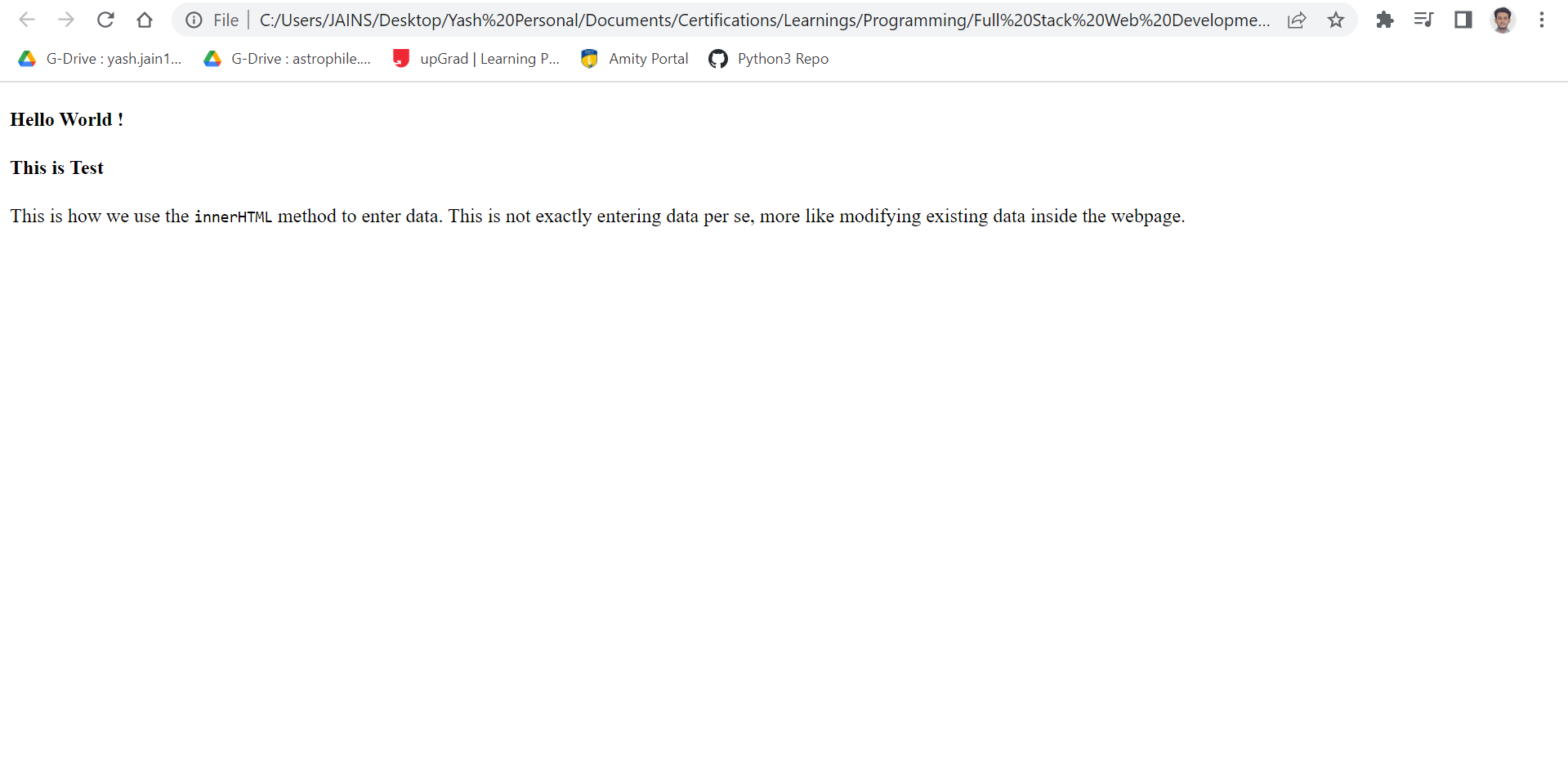
</script>

<p> This is how we use the <code> innerHTML</code> method to enter data. This is not exactly entering data per se, more like modifying existing data inside the webpage.</p>

</body>

</html>

Let us see the output now.



We can see that for the ID attribute, it only modified the contents of the first one it found. This is because ID is unique to each element. We have an additional function – the console.log() function.

Console.log() – This function prints the string inside the parenthesis to the console directly. The console is not the screen on the browsers, but the JavaScript console running at its backend.

It is really helpful in checking bugs to your javascript code, and also for making little notes, that you don’t users to see. It uses the following syntax –

console.log(“string”)

Now, let us see a small example.

<!DOCTYPE html>

<html lang='en'>

<head> <title> The console.log() Function </title> </head>

<body style="width:75%">

<h1> Using the <code>console.log()</code> Function</h1>

<p> The <code>console.log() </code> function helps you identify errors with your javascript code, or help you enter some notes on what you are doing. If you see here, we will accidently forget using one double-quotation mark " in the javascript code.

<script> console.log(This is HTML") </script>

<p> When you will go to the console by right-clicking on the page and clicking on inspect, you will see an error. After correction, the error won't come. The following image depicts error on the console - </p>

<script> console.log("This is HTML"); </script>

<img src="../../Images/error.png" width=50%>

</body>

</html>

Now, let us see the output for this code.

Graphical user interface, text, application, email

Description automatically generated

This is how we can make use of the console to debug our code. It won’t display on the screen since it will be going on the backend, but it is really helpful for programmers.

# Variables in JavaScript

Now, let us start learning about variables in JavaScript. First, we need to learn what they are and why are they used.

An important part of programming is learning how you can save data. Because by saving data, you can reuse it and give you program that's impression that's kind of intelligent and knows the user very well.

In JavaScript, data is stored in what we call **Variables**, and it's very easy to use variables in your programming. The only important part is that you need to tell the computer very specifically, that I need you to save something for me.

We start by using the keyword var. It is followed by our identifier, and an

assignment operator (=) and the value it is going to store. It follows the following syntax –

var varname = value;

This statement means that value is stored in a variable with the name varname.

The word varname is our identifier. It is the alias which points to a certain location in the memory where our value is stored.

You can see that the var word is highlighted. This is because it is a special word in JavaScript. Such words are called keywords. Let us know about them.

**Keywords vs Variables**

The important thing is that identifier should be special to you, but not special to the computer. How does this work?

When you declare a variable, you're basically telling the computer, let's stop talking in human talk and let's talk computer talk. However, when you use a name that is special to the computer, then it will not see it as a variable name. You cannot name a variable as var.

Such words are reserved words in a programming language, called **Keywords** and their meaning is already defined by the language. So whenever you are using them, you are asking the language to use the property of that keyword.

**Rules for Declaring Variables**

While naming our variables, we need to follow some basic rules. These rules are listed below –

* They can contain letters, digits, underscores (\_) and dollar signs ($)
* They cannot start with a digit
* They are case-sensitive. The identifiers name and Name are different.
* They should be pneumonic, i.e. they should represent something in the real world. This is not compulsory but suggested.
* They should not be declared if they are not used in the program.

Pneumonic variables are much better to understand when you review your code. If you name your variable xyz, then it won’t show an error, but you won’t be able to understand it once you review your code.

Let us see an example.

var cust\_name=“Yash”;

var xysfk=“Yash”;

The first identifier is representing a real word entity – customer name. The second identifier is not representing it per se. It is just about semantics.

Let us see an example now.



This code will take our name as input, and then return it as defined in the var.js file. Let us see the output now.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

See? This is how we use variables to generate interactive content in our webpage. They will be used extensively further. Now, let us move towards Data Types in JavaScript – another important component.

# Data Types in JavaScript

Once you realize that computers store all of the information using variables, the next step is to start to learn about the different data types used in the JavaScript programming language.

In computer programming languages, variables tend to have a single type. In JavaScript, that's fine, a variable can only be one type, but throughout the course of the program, it can switch from being a program to characters, back to a number to something else that's completely complex. So let's talk about what these types are in JavaScript and how we can represent them.

There are essentially 3 data types in JavaScript –

* Numbers – Any numeric value with or without decimal
* Strings – Alphanumeric values enclosed within double quotations “”
* Booleans – Values that are only True (1) or False (0).

Let us see an example of such variables.

var radius=14;

var pi=3.14;

var str= “Area”;

var bool= True;

document.write(‘The radius is : ’,radius);

document.write(‘The value of pi is: ’,pi);

document.write(‘The string is: ’,str);

document.write(‘The correct value is: ’,bool);

Let us see the output for this –



See? This is how we use the variables of different data types in JavaScript and print them out to HTML.

Apart from these 3, we also have 2 additional data types –

* Arrays – List of values having same data type
* Object – Objects retrieved from OOPS or from DOM

These are created using different keyword called const, but we will study them later, because these are not data types, but rather, they are data structures.

First, let us see how we can use variables in JavaScript.

# Operators and Expressions in JavaScript

We have been using statements to execute our JavaScript code. Every time you saw a line that ended with a semicolon, we were writing a statement.

Statements often have what we call ***Expressions***, or things that can be evaluated. So expressions produce values. They might produce a number, or a string, but many times they produce boolean values. So let us see all the different types of operators that we can use in JavaScript to produce these types of expression values.

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Type | Purpose | Example |
| = | Assignment | Storing values | x =12; y =13; |
| + | Arithmetic | Addition | z = x+y |
| - | Arithmetic | Subtraction | z = x-y |
| \* | Arithmetic | Multiplication | z = x\*y |
| / | Arithmetic | Division | z = x/y |
| \*\* | Arithmetic | Exponential | z = x\*\*y |
| % | Arithmetic | Modulus | z = x%y |
| == | Comparison | Checking equality | x==y |
| != | Comparison | Check inequality | x!=y |
| >= | Comparison | Check inequality | x>=y |
| <= | Comparison | Check inequality | X<=y |
| && | Logical | AND operation | z = (x>2&&y<2) |
| || | Logical | OR operation | z = (x>2||y<2) |
| ! | Logical | NOT operation | z = (x!=y) |

The logical operations cannot be done without the comparison operations, except for the NOT (!) operator.

Let us see an example using all the operators.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title> Operators and Expressions in JavaScript</title>

</head>

<body style="font-size:130%">

<h1> Operators and Expressions in JavaScript</h1>

<p> We will be calculating some arithmetic operations and some logical operations here.</p>

<p><script src="../../JS/operators.js"></script></p>

</body>

</html>

var x=3;

var y=4;

var sum=x+y;

var diff=x-y;

var prod=x\*y;

var quo=y/x;

var rem=y%x;

var ex=x\*\*y;

document.write('The variables are : <code>',x,'</code> and <code>',y,'</code>');

document.write('<h3>Arithmetic Operations</h3>');

document.write('Sum is: <code>',sum,'</code><br>');

document.write('Difference is: <code>',diff,'</code><br>');

document.write('Product is: <code>',prod,'</code><br>');

document.write('Quotient is: <code>',quo,'</code><br>');

document.write('Remainder is: <code>',rem,'</code><br>');

document.write('Exponential is: <code>',ex,'</code>');

var logand=(x>2&&y<2);

var logor=(x>2||y<2);

var lognot=(x!=y);

document.write('<h3>Logical Operations</h3>');

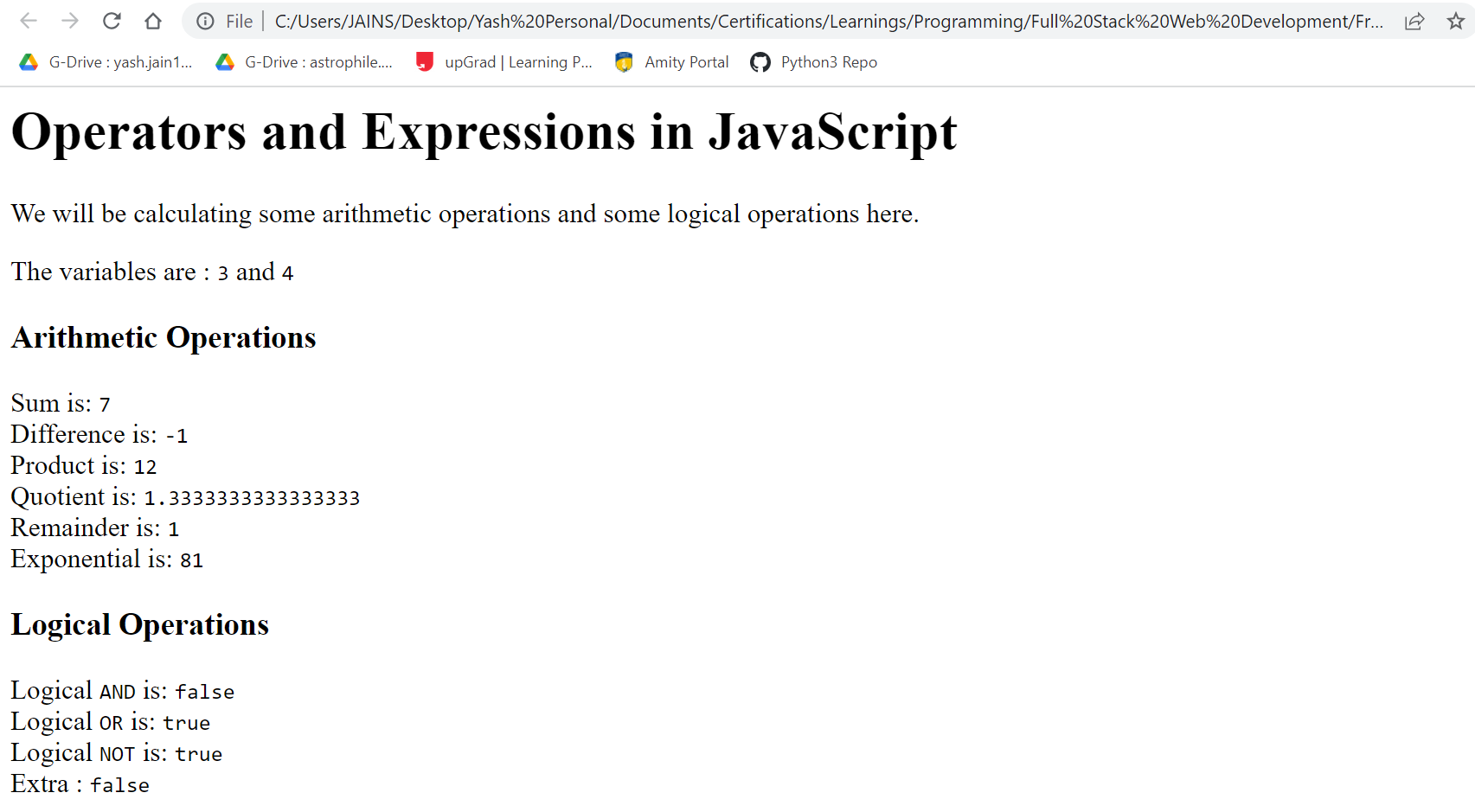
document.write('Logical <code>AND</code> is: <code>',logand,'</code><br>');

document.write('Logical <code>OR</code> is: <code>',logor,'</code><br>');

document.write('Logical <code>NOT</code> is: <code>',lognot,'</code><br>');

document.write('Extra : <code>',!x,'</code>');

Now, let us see the output for this code.



This is how we use operators and expressions in JavaScript to get interactive and dynamic content. This is where the use of APIs becomes crucial, as depicted in the js file code.

Apart from these operators, we have 3 new operators in JavaScript –

* +=n – This is used on just one operand, and increases the value by n.
* -=n – This is used on just one operand, and decreases the value by n.
* === – This is a comparison operator used for strict equality.

Let us see some examples for them.

x=3;

y=‘3’;

x+=5; // this will increase x by 5. Hence x will be 8.

x-=2; // this will decrease x by 2. Hence x will be 6.

x+=y; // this will treat both x and y as strings and will join them.

console.log(x);

console.log(y==3); // this will return True, even though the data types are different.

console.log(y===3); // this will return False. It checked for both value and type.

Output:

63

true

false

Hence, this is how operators are used in JavaScript. Each and every operator has its own limitations.

The division and modulus operators can only be used on numerical values, whereas the addition and multiplication can also be used on strings. They will join strings and repeat strings respectively. Therefore, it is important to try and learn about their limitations.

This was all about operators and expressions.