**What is JavaScript?**

JavaScript is a ***scripting***, ***lightweight,*** ***interpreted*** ***OOP*** programming language that has ***first-class functions*** and allows you to implement complex features on web pages to make them do more than just sit there and display static information for you to look at.

* ***Lightweight*** - has small memory footprint and easy to implement.
* ***Interpreted*** - no compilation and instructions executed directly.
* ***Object Orientated Programming*** - modelled around objects.
* ***First-class functions*** – functions as values, meaning you can assign a function to a variable.
* ***Scripting*** – instructions are written for a runtime environment like cmd or even web browser.

It is used for ***client side*** web development.

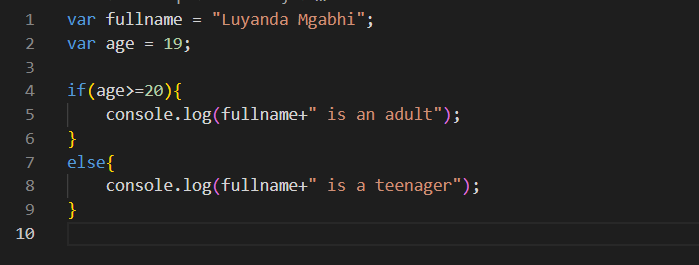
* ***Native JavaScript*** – pure JavaScript with no frameworks.
* ***JQuery Framework*** – lets you modify/ manipulate the structure on the client side.
* ***AngularJS, ReactJs Frameworks*** – advanced flying side application frameworks.

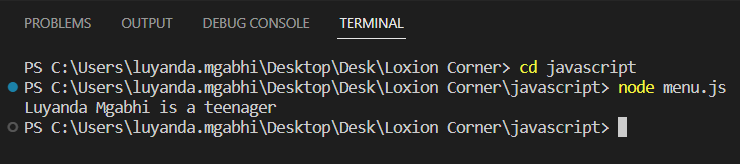
It is also used for ***server side*** web development.

* ***NodeJS Framework*** – lets you modify/ manipulate the structure on the server side.

JavaScript doesn’t have the concept of variable types like in Java, for example when declaring an int or string type variable in Java the below syntax would be used:

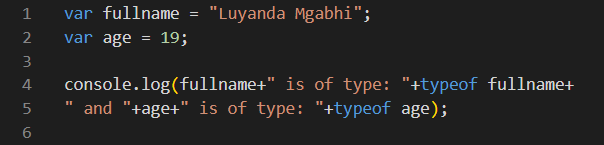
* Java syntax: int age = 25;/ string name = “Cyco”;
* JavaScript: var age = 25;/ var name = “Cyco”;
* JavaScript uses the ***var*** keyword to declare any variable type.

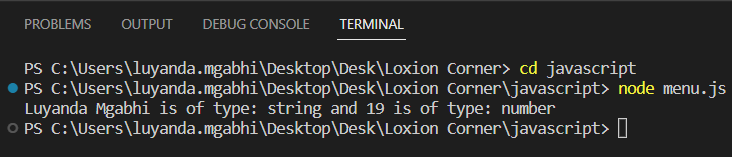




There’s no ***scoping*** information in variable declarations like in java where you’d specify if a field is ***public, protected or private***.

The ***typeof operator*** is used to return the type of a variable or value as shown below:

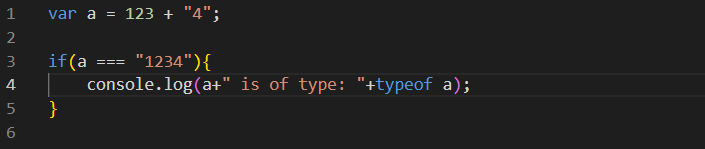


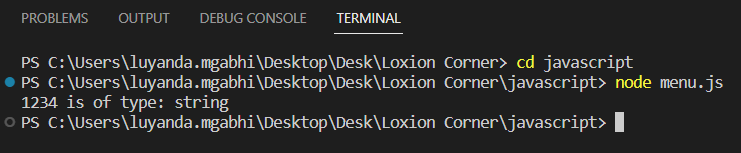


**Note:** The value ***null*** is of type ***object***.

JavaScript also has what is called ***Type Coercion*** which is actually the ability to concatenate string values with other types value like number.

* For example, if you have a variable ***a = 123 + “4”;***
* JavaScript will first convert the ***123*** number type to a string type and then appends ***“4”*** to it, hence return a final value of a string ***a = “1234”;***
* The ***=== operator*** is used to compare if two values are equal to each other.
* The ***== operator*** works for types number, string, boolean, null, undefined.
* If you compare a string variable ***a = “10”*** to an unassigned number ***10*** using the ***== operator***, JavaScript interprets this as if you are compare two strings because it chooses the variable type of an unassigned value type.
* Similarly, if you compare a number variable ***a = 10*** to an unassigned string, the same applies.
* The above doesn’t apply to the ***=== operator***, as the above is a bug from previous rollout.





Important to note is that is you declared and defined a variable ***a = 10 or “10” or true*** of types ***number, string and boolean***, then have an if statement ***if(a){do something;}***, this will turn out to be and execute the if statement.

Unless if the ***number value is 0, or string value is empty or boolean value is false***, then the ***if(a){do something;}*** condition won’t be executed.

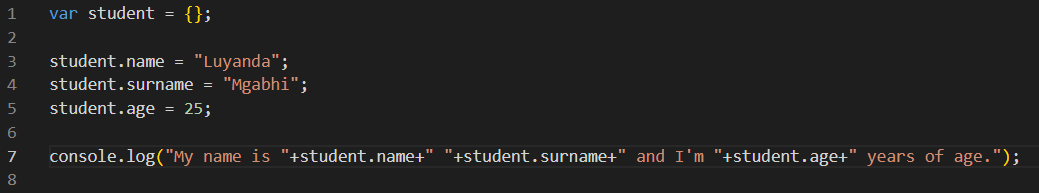
The ***if(a){do something;}*** statement will also not execute if a variable is assigned a value of ***null*** or ***undefined***.

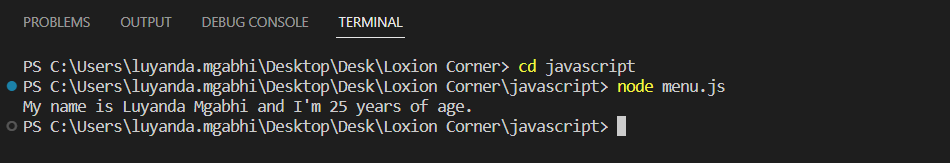
To create objects in JavaScript, classes are not needed like in Java. Objects are created by declaring an object variable ***var student = {};***

An ***object*** is essentially a collection of ***data*** and ***functionalities***, where data is referred as ***properties*** and functionalities as ***methods***.

To add properties to the ***student object*** above, the below would be done:

* To add a name property – student.name = “Luyanda”;
* To add a surname property – student.surname = “Mgabhi”;
* To add an age property – student.age = “25”;





Objects can also be defined ***inline*** as follows:

***var student = {***

***"name": "Luyanda",***

***"surname": "Mgabhi",***

***"age": 25***

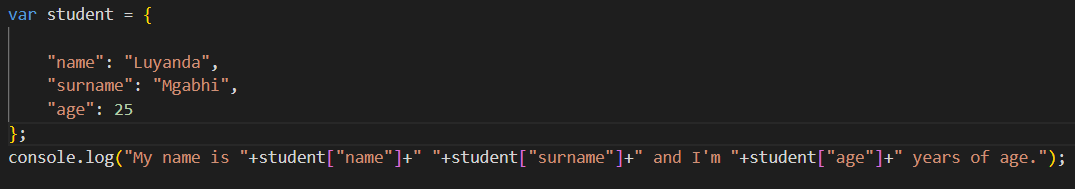
***};***

The above would print out exactly the same result as before.

Also to note, accessing properties that don’t exist within an object, the result is returned as undefined.

There are two ways to access object properties and methods, one is the above which is the ***dot notation***, the other is ***square bracket [] notation***.

* To use the square bracket notation, you can say ***student[“name”].***
* So the object property is inserted in a square bracket as a string.

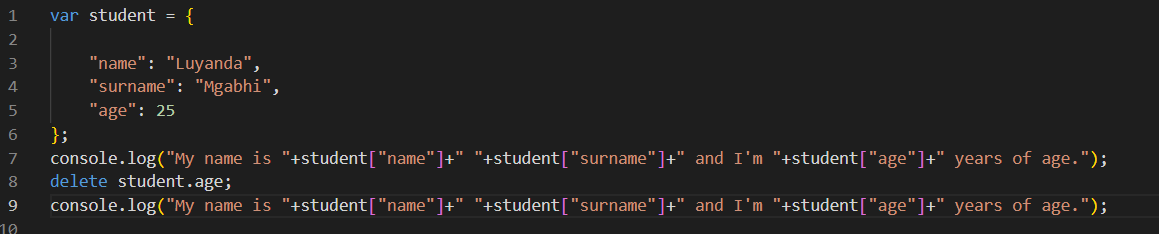


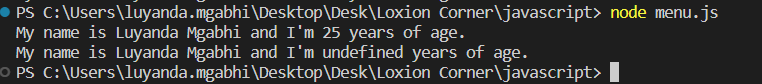
The results would be exactly the same as previously.

The use of the two notations differ for different scenarios as follows:

* The square bracket notation is used when property name is a ***reserved word***/ ***invalid identifier***.
* An example of an invalid identifier would be something like this ***student.1 = “Luyanda”*** or for inline object ***“1”: “Luyanda”***.
* What the above would mean is that, you can’t have a number as an identifier, it has to begin with a letter when using the ***dot notation***.
* The only way you can access an invalid identifier is by using a square bracket notation as follows, ***console.log(student[“1”]);*** This would print the value “Luyanda”.
* Another scenario is when you have a dynamic property name, this means you wouldn’t know what the property name is and relying on a user input.
* The code to this would be as follows, ***var propertyName = <user input>;***, then to access this you would say ***console.log(student[propertyName]);***
* This would print whatever object property the user passed.

To delete or remove object properties the ***delete*** keyword is used as follows: ***delete student.age;***



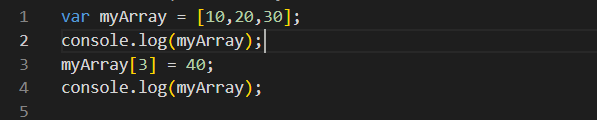


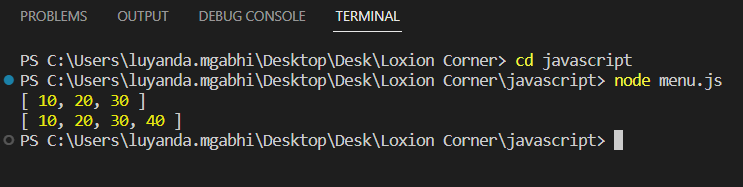
When you try to access or print a deleted property, it return an undefined value as shown in the output above.

To define an array in JavaScript the following syntax is used: ***var myArray = [10, 20, 30];***

To access the array elements, ***myArray[i]*** is used where ***i*** is the index value of the array element.

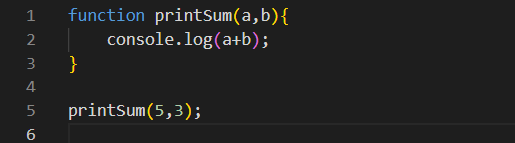
To add a new 4th element to an existing array of size 3, ***myArray[3] = 40***; is used.





To create a function in JavaScript, the following syntax is used: ***function myFunction(a,b){function code to be executed or returned}***.

To call a function, you use ***<function name>(<argument 1>, <argument 2>);***





If you provide less function arguments at runtime then the function returns undefined for the missing argument.

If you provide more function arguments at runtime then the function simply ignores the extra argument/ arguments and executes the function code as normal.

***Function overload*** doesn’t exist in JavaScript, that is you can’t use the same function name for more than one function even if they have different number of arguments.

Functions can also be assigned to variables as follows: ***var f = function printSum(a,b){function code};*** then to call the function you just call the variable as a function with arguments if they exist as follows: ***f(a,b);*** . You can also have the above without the function name ***printSum*** in the above case, ***var f = function (a,b){function code};*** . This is known as ***anonymous function expressions***.

To add functions to object, ***var myObject.printSum = function (a,b){function code;};***

To call the function using the object, ***myObject.printSum(a,b);***

For in JavaScript is the same as in Java.

***The myArray.push(element)*** method is used to add elements.

The ***myArray.pop()*** method is used to delete the last element of the array.

The ***myArray.shift()*** method is used to delete the first element of the array.

The ***myArray.unshift(element)*** method is used to add an element at the beginning of the array.

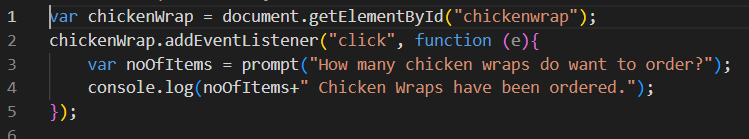
To add a JavaScript code to HTML elements, the ***<script src=”JavaScript file name/path”></script>*** tags are used

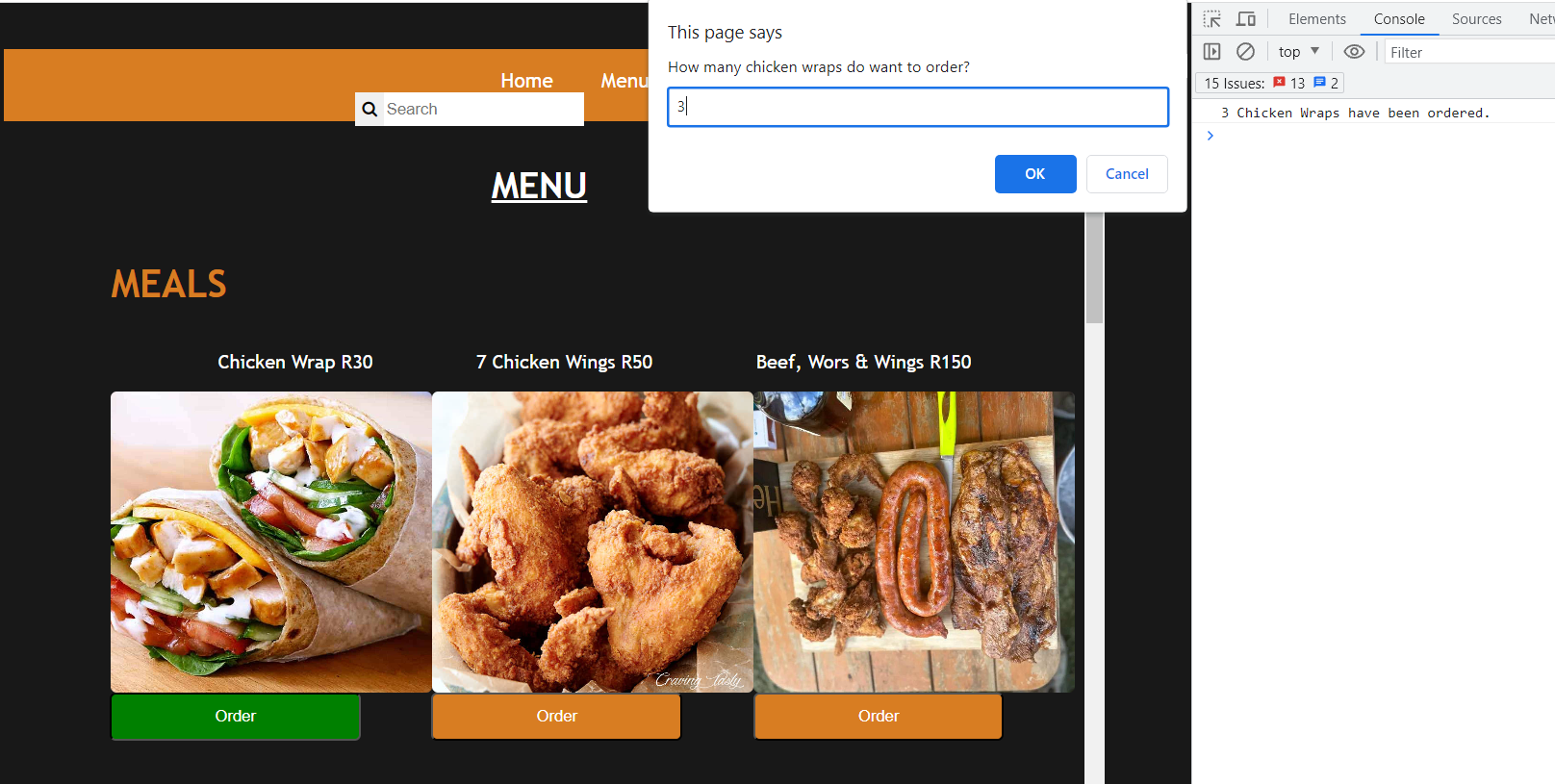
To reference an HTML element like a button in a JavaScript file, you can declare a variable/ constant as follows ***var/const myBtn = document.getElementById(<id name in the html file>);***

To add an event or make the button do something when clicked, the ***myBtn.addEventListener(“click”, function(e){do something});*** is used.

To prompt a user and store the user response, the ***var response = prompt(“enter your response”);*** Syntax is used.

The response can be then used for some sort of processing.

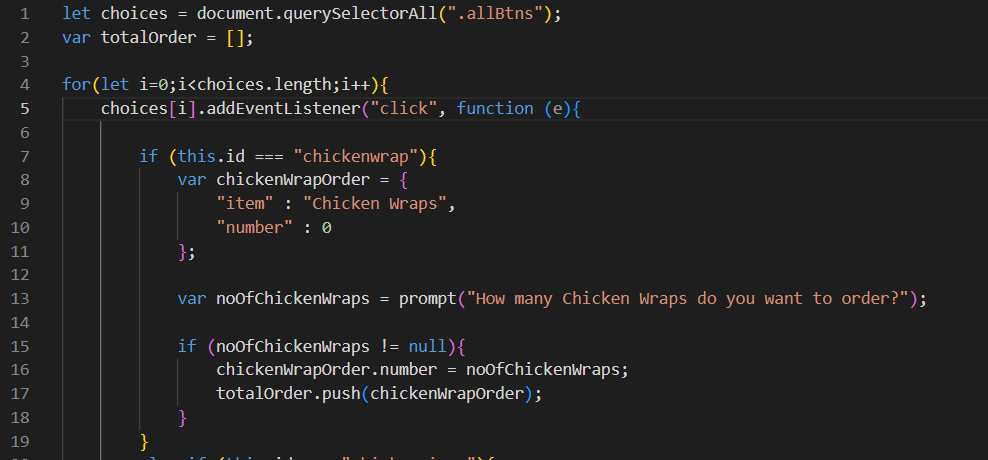


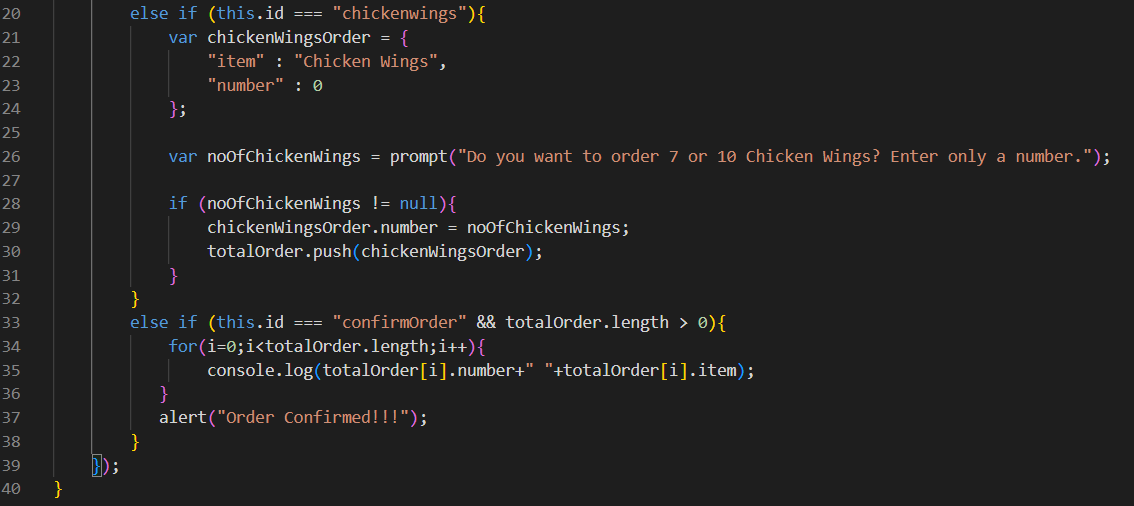


If there’s multiple buttons a user can choose from, the following syntax is used to get the button Id clicked: ***let choices = document.querySelectorAll(".allBtns");***

The above syntax works as follows, it checks and selects all the buttons with the ***HTML class=”allBtns”*** and stores them in an array sort of form which is assigned to a variable ***choices*** in the above example.

An eventlistener method is used within a for loop to access the buttons when clicked as shown below:





The function ***isNaN(argument***) is used to check if the argument is a number, and returns a boolean value.

To change a string number value to number value, the ***+*** is added in front of the string number value.

To update HTML text using JavaScript, ***document.getElementById("cartprice").innerHTML = "R"+totalPrice;*** syntax is used. Where ***“cartprice”*** is the text element being updated and ***“R”+totalPrice*** is the value of the update.