



Project 06

“JAVASCRIPTS BASICS”



SUBMITTED BY:
MANASWI M. PATIL



INDEX

Sr no.	Topic	Pg no.
1.	Introduction	3
2.	Variables in JavaScript	5
3.	Variables in JavaScript	6
4.	JavaScript Operators and Expressions	8
5.	Loops in JavaScript	12
6.	JavaScript Features	13
7.	Conclusion	20

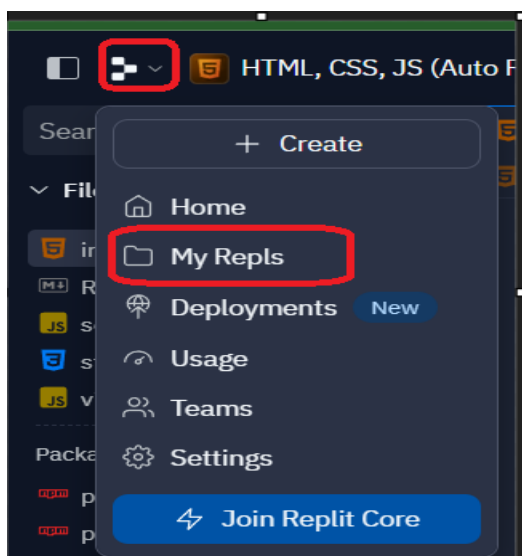


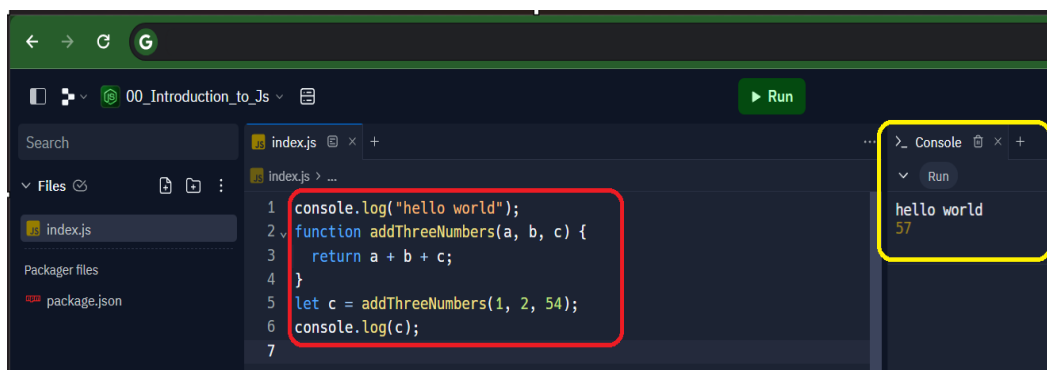
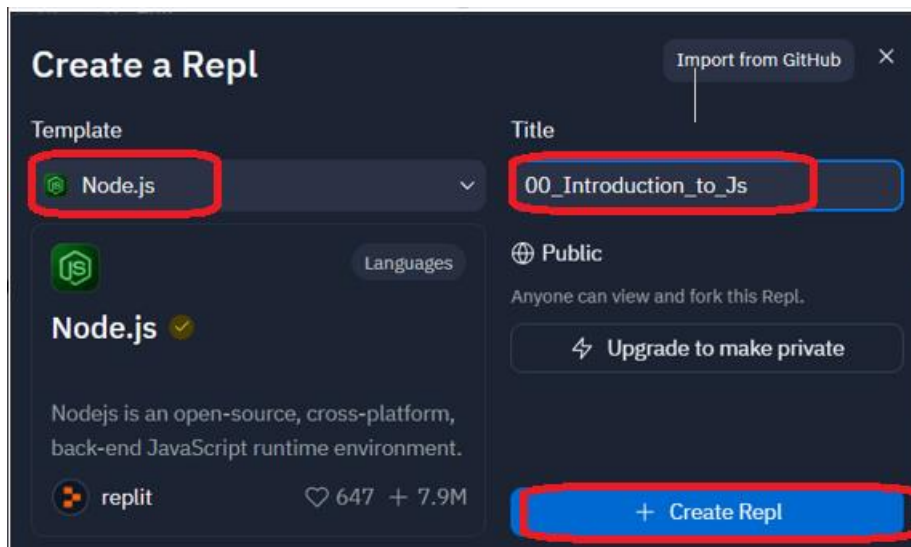
1. INTRODUCTION

JavaScript is a programming language that enables interactive and dynamic content on websites. It's commonly used to enhance the user experience by allowing developers to create responsive and interactive elements. JavaScript runs in web browsers and is essential for building features like forms, animations, and updates on web pages without requiring a page refresh. It's a versatile language that plays a crucial role in web development. JavaScript allows developers to create dynamic and interactive elements on web pages, improving the user experience. It empowers client-side scripting, enabling browsers to execute scripts locally on users' devices, reducing server load. It enables the manipulation of HTML and CSS elements in real-time, allowing for the modification of page content without requiring a full page reload. It ensures compatibility across various browsers, making it a versatile language for web development. JavaScript helps in creating web applications that work consistently across different browsers, ensuring a seamless user experience.

Execution of JavaScript using Replit in chrome browser

- open browser → type 'replit.com' → sign up your account or login → click on my repls → new folder → create folder → click on folder → click new repls → search in template latest version of 'node.js' → title '00_Introduction_to_Js' → new repls.





WHAT IS ECMAScript

ECMAScript (ES) is a scripting language specification that serves as the foundation for JavaScript. It defines the **rules and guidelines** that JavaScript engines follow to ensure consistency and interoperability.

How to commenting?

- i. In JavaScript you used **'ctrl+/'** to commenting the line

```
// console.log('Operators in Js')
// Arithmetic operators
```

- ii. And for opening **multiline commenting** you used **'/*'** and for closing **'*/'**.

The multiline comment is used to provide a block of text that serves as documentation or comments about the code. It's common to use multiline comments for documenting functions, describing the purpose of a section of code, or providing information about the overall structure of a file.

```
/*console.log('Operators in Js')
Arithmetic operators*/
```

2. Variables in JavaScript

1. Also used '**var**' to declared variables in JavaScript. we can used var to redeclared. var was the primary keyword used for variable declaration. Variables declared with var are function-scoped or globally-scoped, and they can be redeclared.

```
var x = 5;
var x = 10; // This is allowed, and it reassigns the value of 'x'
```

2. In JavaScript we used '**Let**' to declare variables.(blocked scoped variable)but we cannot use it to **redeclared variable** because it will throw errors.

```
index.js  ×  +
index.js > ...
1 console.log("variable declaration")
2 var a=45;
3 var a= "m";
4 let b= "manu";
5 let b= 50;
```

```
SyntaxError: Identifier 'b' has already been declared
    at internal/compileFunction (node:internal/umy77:19)
```

```
index.js  ×  +
index.js > ...
1 console.log("variable declaration")
2 var a=45;
3 var a= "m";
4 let b= "manu";
5 {
6   let b='this'
7   console.log(b)
8 }
9 console.log(b)
10
```

Blocked scope

```
variable declaration
this
manu
```

3. If variable value will remain same then used '**const**' function. which are block-scoped and cannot be reassigned after declaration. They must be assigned a value at the time of declaration.

```
const z = 5;
// z = 10; // This would result in an error, as 'const' variables
cannot be reassigned
```



3. Datatypes in JavaScript

Primitive and Non-Primitive

There are total **seven** types of data types:

Types	Description
Null	Represents the intentional absence of any object value.
Number	Represents numeric values, both integers and floating-point numbers.
String	Represents sequences of characters enclosed in single or double quotes.
Symbol	Represents unique identifiers. Symbols are often used as keys in objects to avoid naming conflicts.
Undefined	Represents a variable that has been declared but not assigned a value.
Boolean	Represents either true or false, typically used for logical operations.
BigInt	Represents integers of arbitrary precision, useful when dealing with extremely large numbers.

These primitive data types are immutable, meaning their values cannot be changed after they are created. In contrast, objects (including arrays and functions) are considered non-primitive data types and are mutable.

Objects in JavaScript

How to create an Object?

Objects can be created using the curly braces ‘{}’ and adding key-value pairs inside.

```
const person = {  
  name: "mau",  
  age: 23,  
};
```

Accessing Values:

You can access values in an object using the ‘dot notation’ or ‘square bracket notation.’

```
const person = {  
  name: "mau",  
  age: 23,  
};  
console.log(person.name); // Output: mau  
console.log(person["age"]); // Output: 23
```

Adding and Modifying Properties:

You can add new properties or modify existing ones in an object.



```
person.gender = 'female' ;  
person.age = 23;  
  
console.log(person.gender); // Output: female  
console.log(person.age);    // Output: 23
```

Nested Objects:

Objects can contain other objects, creating a nested structure.

```
let address = {  
  city: 'New York',  
  zipCode: '10001',  
};  
  
person.address = address;  
  
console.log(person.address.city); // Output: New York
```

Methods:

You can include functions as properties in an object, and they are referred to as methods.

```
let car = {  
  brand: 'Toyota',  
  model: 'Camry',  
  start: function() {  
    console.log('Engine started');  
  },  
};  
  
car.start(); // Output: Engine started
```

Removing Properties:

You can delete properties from an object using the delete keyword.



4. JAVASCRIPTS OPERATORS AND EXPRESSSIONS

I. **Operators** are symbols that perform operations on operands. Operands can be variables, values, or expressions.

a. Arithmetic operators

Arithmetic operation are used to perform mathematical operations.

```
index.js > ...
1 console.log('Operators in Js')
2 // Arithmetic operators
3 let a=14;
4 let b=4;
5 console.log("a+b=",a+b)
6 console.log("a-b=",a-b)
7 console.log("a*b=",a*b)
8 console.log("a/b=",a/b)
9 console.log("a%b=",a%b)
10 console.log("a++=",a++)
11 console.log("--a=",--a)
12 console.log("a--=",a--)
13 console.log("++a=",++a)
```

```
Run
Operators in Js
a+b= 18
a-b= 10
a*b= 56
a/b= 3.5
a%b= 2
a++= 14
--a= 14
a--= 14
++a= 14
```

b. Assignment operators

In JavaScript, assignment operators are used to assign values to variables. They are shorthand notations that combine the assignment (=) with another operation.

```
// assignment opeartors
let x = 3;
let y=7;
console.log(x += 3);
console.log(x *= 3);
console.log(x %= 3);
console.log(x -= 3);
```

```
Run
Run
6
18
0
-3
```

c. Comparison operators

Comparison operators in JavaScript are used to compare two values and return a Boolean result indicating the relationship between them.

```
// comparison operator
let x = 5;
let y = 7;
console.log(x == y); //equal to
console.log(x != y); // not equal
console.log(x === y); //strict equal
console.log(x !== y); // strict not equal
console.log(x >= y); //greater than equal to
console.log(x <= y); // less than equal to
console.log(x > y); //greater than
console.log(x < y); //less than
```

```
Formatter Form
Run
false
true
false
true
false
true
false
true
```

d. Logical operators

Logical operators in JavaScript are used to perform logical operations on values.



```
// logical operators
let x=5;
let y=6;
console.log(x<y && x==5)
console.log(x>y ||x==5)
console.log(!true)
console.log(!false)
```

Run

```
true
true
false
true
```

e. Ternary operators

Evaluates the **condition** and **execute the block of code** based on the condition.

II. JavaScript Functions

In JavaScript, the **function keyword** is used to declare a function. javascript function is a block of code used to declared a task. Functions holds the input value.

```
index.js > ...
1 function itemsAvg(x,y) {
2   return 1 + (x + y) / 2;
3 }
4 let a = 1;
5 let b = 2;
6 let c = 3;
7 console.log("items ", itemsAvg(x, y));
8 console.log("items",1 + (x + y) / 2);
9 |
```

Syntax:

```
function functionName(parameters) {
  // code to be executed
}
```

- ❖ **function:** This is the keyword that tells JavaScript that you are declaring a function.
- ❖ **Function Name:** This is the name of the function. You can choose a meaningful name that describes what the function does. It follows the same rules as variable names.
- ❖ **parameters:** These are optional. They are placeholders for values that the function will receive when it is called. You can have zero or more parameters.
- ❖ **{}**: This is the function body, where you write the code that the function will execute.
- ❖ **Call ():** With the **call()** method, you can write a method that can be used on different objects.
- ❖ **Apply():** With the **apply()** method, you can write a method that can be used on different objects.
- ❖ With the **bind()** method, an object can borrow a method from another object.

III. Conditional statements:

i. If statement

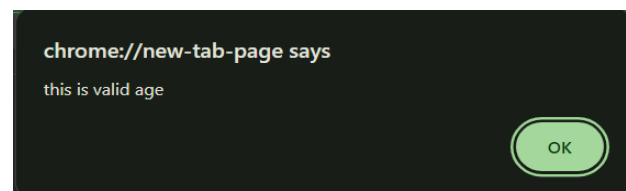
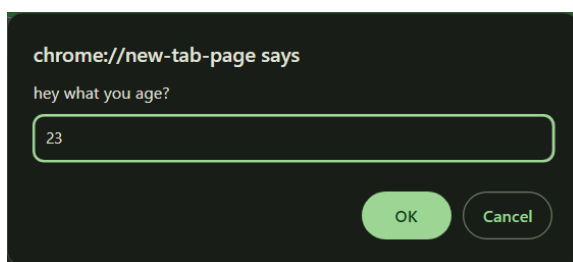


Open chrome browser → type code in console → enter to run → or used replit → in replit window to run prompt used `'const prompt=require ("prompt-sync")();'`.

Prompt: You asked question in this.

Alert : It will give the response.

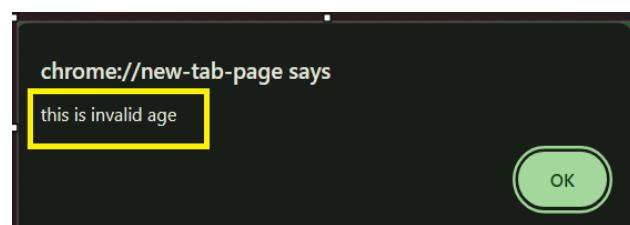
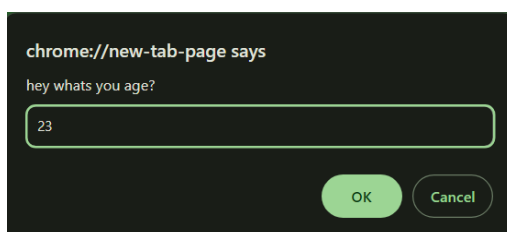
```
> let a = prompt("hey what you age?");  
a = Number.parseInt(a); //convert string to number  
if(a>0) {  
    alert("this is valid age");  
}  
undefined
```



ii. if else.. statement

If the condition specified in the if statement is true, the block of code inside the curly braces immediately following the if statement will be executed. If the condition is false, the block of code inside the else statement (if present) will be executed.

```
> let a = prompt("hey whats you age?");  
a = Number.parseInt(a); //convert string to number  
if (a < 0) {  
    alert("this is valid age");  
} else {  
    alert("this is invalid age");  
}
```



iii. if else.. if else statement

The **if...else if...else** statement allows you to handle multiple conditions. Each if and else if block contains a condition. If the condition in the if statement is true, the corresponding block of code is executed, and the rest of the else if and else blocks are skipped. If the condition in the if statement is false, the program moves on to the next else if statement. If none of the conditions are true, the code inside the else block (if present) is executed.



iv. switch statement

In JavaScript, the switch statement provides a way to handle multiple conditions in a concise and readable manner.

```
1 let day = 4;
2 let dayName;
3
4 switch (day) {
5   case 1:
6     dayName = "Monday";
7     break;
8   case 2:
9     dayName = "Tuesday";
10    break;
11   case 3:
12     dayName = "Wednesday";
13     break;
14   default:
15     dayName = "Invalid day";
16 }
17
18 console.log(dayName);
```



IV. JavaScript(JS) Const

- ❖ Variables defined with const cannot be **Redeclared**
- ❖ Variables defined with const cannot be **Reassigned**
- ❖ Variables defined with const have **Block Scope**

When you used JavaScript const?

Always declare a variable with 'const' when you know that the value **should not be changed**.

Used 'const' when you declare "A new Array, A new Object, A new Function, A new RegExp."

This is good because of this features:

- let and const have **block scope**.
- let and const can **not be redeclared**.
- let and const must be **declared before use**.
- let and const does **not bind to this**.
- let and const are **not hoisted**.

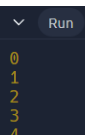


5. LOOPS IN JAVASCRIPT

a. for Loop:

The for loop is used when the **number of iterations is known in advance**. It consists of three parts: initialization, condition, and iteration.

```
for (let i = 0; i < 5; i++) {  
  console.log(i);  
}
```



b. while Loop:

The while loop **repeats a block of code** while a specified condition is true.

```
let i = 0;  
while (i < 5) {  
  console.log(i);  
  i++;  
}
```



c. do...while Loop:

The do...while loop is **similar to the while loop**, but it guarantees that the code block is executed at least once before checking the condition.


```
let i = 0;  
do {  
  console.log(i);  
  i++;  
} while (i < 5);
```



d. for...in Loop:

The for...in loop **iterates over the properties of an object**.

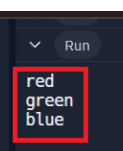
```
const person = { name: "John", age: 30, job: "developer" };  
for (let key in person) {  
  console.log(key, person[key]);  
}
```



e. for...of Loop:

The for...of loop is used to iterate over iterable objects (arrays, strings, etc.).

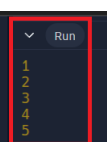
```
const colors = ["red", "green", "blue"];  
for (let color of colors) {  
  console.log(color);  
}
```



f. forEach Method:

For arrays, you can use the forEach method to iterate over each element.

```
const numbers = [1, 2, 3, 4, 5];  
numbers.forEach(function (number) {  
  console.log(number);  
});
```





6. JAVASCRIPTS FEATURES

I. JavaScript Set

You can create a new Set using the Set constructor.

Syntax:

```
let mySet = new Set();
```

How to add element using set?

You can add elements to a set using the **add method**.

```
let mySet = new Set();  
mySet.add(1);  
mySet.add("Hello");  
mySet.add(true);  
mySet.add({ name: "John" });
```

- **new Set():** Creates a new Set
- **add():** Adds a new element to the Set
- **delete():** Removes an element from a Set
- **has():** Returns true if a value exists in the Set
- **forEach():** Invokes a callback for each element in the Set
- **values():** Returns an iterator with all the values in a Set
- **size:** Returns the number of elements in a Set

II. JavaScript Map

How to Create a Map?

In JavaScript, the map function is a method that can be called on an array. It is used to create a new array by applying a provided function to each element of an existing array.

- ❖ Passing an Array to new Map()
- ❖ Create a Map and use Map.set()

```
const originalArray = [1, 2, 3, 4, 5];  
  
const newArray = originalArray.map(function(element) {  
    return element * 2;  
});  
console.log(newArray);
```

Run

[2, 4, 6, 8, 10]

- **new Map():** Creates a new Map
- **set():** Sets the value for a key in a Map



- **get():** Gets the value for a key in a Map
- **delete():** Removes a Map element specified by the key
- **has():** Returns true if a key exists in a Map
- **forEach():** Calls a function for each key/value pair in a Map .It takes a **value ,index** array elements.
- **entries():** Returns an iterator with the [key, value] pairs in a Map
- **filter():** filters an array with value that passes a test. creates a new array.
- **Reduced():** Reduces a value of single array.

III. JavaScript Events

JavaScript, events are actions or occurrences that happen in the browser, such as a user clicking a button or a page finishing loading. You can declare JavaScript events by attaching event listeners to HTML elements.

IV. JavaScript Strings

JavaScript, strings are used to represent **text and are one of the primitive data types**. They are sequences of characters enclosed in **single ('), double ("), or backticks (`) quotes**.

a. Declaration:

```
let singleQuoted = 'Hello, World!';  
let doubleQuoted = "JavaScript is awesome!";  
let backticks = `String interpolation with ${variable}`;
```

b. Escape Characters:

You can use escape characters to include special characters within a string.

```
let escapedString = 'This is a single-quoted string with a  
newline:\nNew line.';
```

c. String Concatenation:

You can concatenate strings using the + operator.

```
let firstName = "Mau";  
let lastName = "Patil";  
let fullName = firstName + " " + lastName; // Result: 'Mau Patil'
```

d. String Length:

The length property is used to get the length of a string.



```
let message = 'Hello!';  
console.log(message.length); // Output: 6
```

e. Accessing Characters:

Characters within a string can be accessed using index notation.

```
let word = 'JavaScript';  
console.log(word[5]); // Output: 'c'
```

f. String Methods:

JavaScript provides various methods for manipulating strings. Some common ones include toUpperCase(), toLowerCase(), indexOf(), slice(), and many more.

```
let phrase = "JavaScript is case sensitive";  
console.log(phrase.toUpperCase()); // Output: 'JAVASCRIPT IS CASE SENSITIVE'  
console.log(phrase.indexOf("is")); // Output: 11  
console.log(phrase.slice(0, 10)); // Output: 'JavaScript'
```

g. Template Literals:

Template literals, introduced in ECMAScript 6 (ES6), allow for string interpolation and multiline strings using backticks.

```
let variable = 'awesome';  
let templateString = `JavaScript is ${variable}!`;
```

V. JS BIGINT

BigInt is the second numeric data type in JavaScript (after Number). BIGINT Operators that can be used on a JavaScript **Number can also be used on a BigInt**. With BigInt the total number of supported data types in JavaScript is 8.

- i. String
- ii. Number
- iii. BigInt
- iv. Boolean
- v. Undefined
- vi. Null
- vii. Symbol
- viii. Object

VI. JavaScript JSON



JSON is a format for storing and transporting data. JSON is often used when data is sent from a server to a web page.

Syntax should be:

- ❖ Data is in name/value pairs
- ❖ Data is separated by commas
- ❖ Curly braces hold objects
- ❖ Square brackets hold arrays

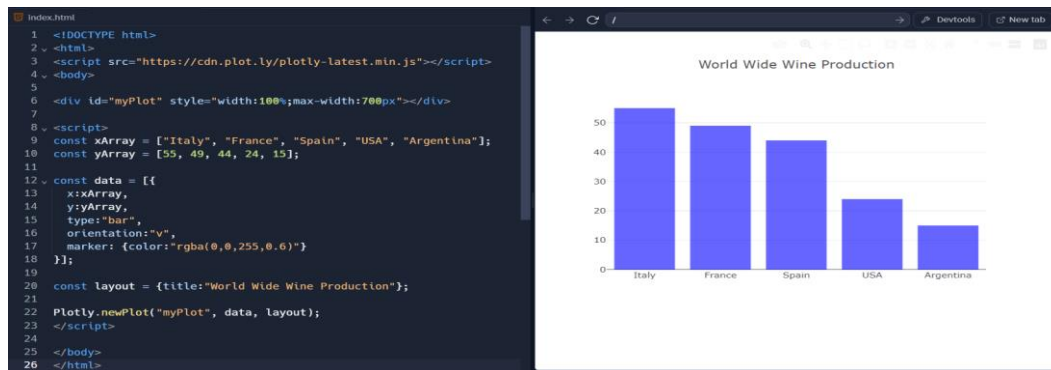
VII. JS TypeOf

You can use the **typeof** operator to find the data type of a JavaScript variable.

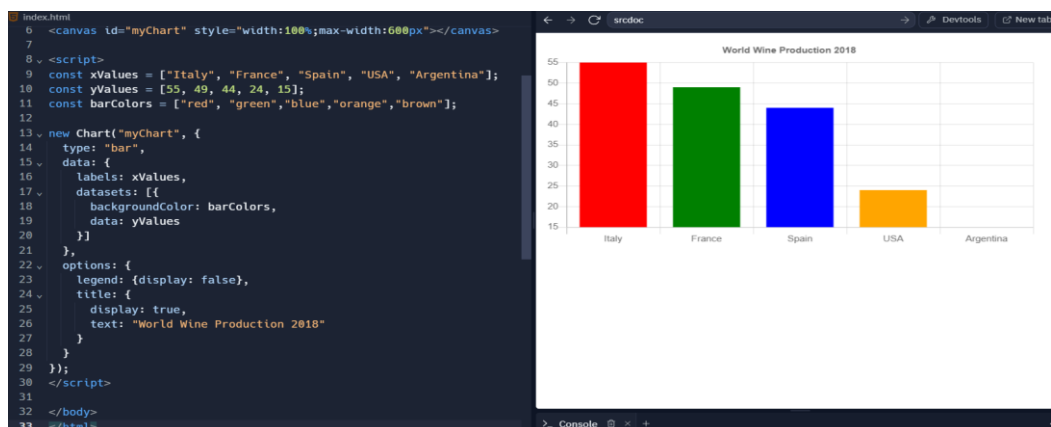
VIII. JS Graphics

JavaScript libraries to use for all kinds of graphs:

1. Plotly.js

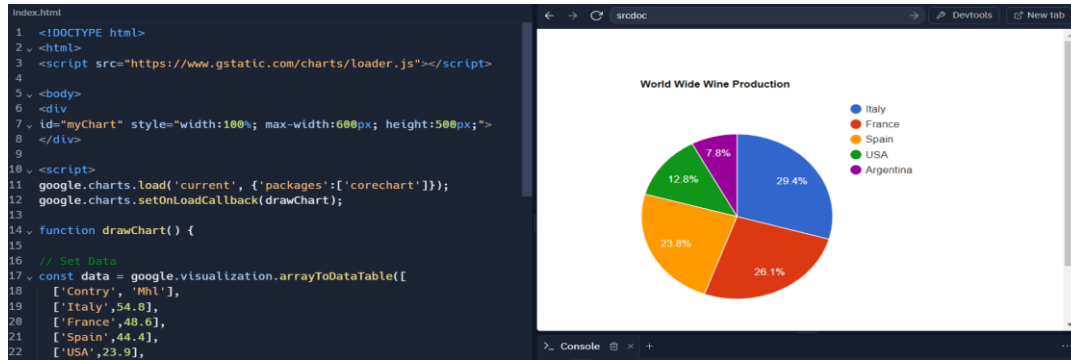


2. Chart.js



3. Google Chart

Run with index.html file



IX. JAVASCRIPT NOTATION

JavaScript Notation refers to the syntax and structure used in JavaScript programming. It includes **various aspects, variables, expressions, statements, objects, and functions.**

- i. **Objects:** Objects are collection of key value pairs.

```
var person = {
  name: 'John',
  age: 30,
  isStudent: false
};
```

- ii. **Variables:** In JavaScript, variables are declared using the **var, let, or const** keyword.

```
var x = 5;
let y = 'Hello';
const PI = 3.14;
```

- iii. **Control Structures:** JavaScript includes control structures like if, else, while, for, and switch.

```
if (condition) {
  // code to run if condition is true
} else {
  // code to run if condition is false
}
```

- iv. **JSON (JavaScript Object Notation):** JSON is a lightweight data interchange format. It uses a syntax similar to JavaScript object notation.



```
{  
  "name": "Alice",  
  "age": 25,  
  "city": "Wonderland"  
}
```

- v. **Arrow Functions (ES6+):** Arrow functions provide a concise syntax for writing functions.

```
const square = (x) => x * x;
```

- vi. **Template Literals (ES6+):** Template literals allow embedding expressions inside string literals.

```
var name = 'World';  
console.log(`Hello, ${name}!`);
```

X. SPECIAL KEYWORDS IN JAVASCRIPT

Keywords	Used
var, let, const	Use for variables
function	Use for declaring function
return	Used inside functions to specify the value to be returned.
if, else if, else	Used for conditional statements.
switch, case, default	Used for multi-way branching based on the value of an expression.
for, while, do-while	Used for loop constructs.
break, continue	Break is used to exit a loop or switch statement. continue is used to skip the rest of the loop body for the current iteration.
try, catch, finally	Used for exception handling. try defines a block of code to be tested for errors, catch specifies a block of code to be executed if an error occurs, and finally specifies a block of code to be executed regardless of the try/catch result.
throw	Used to throw a user-defined exception.



new	Used to create an instance of an object or a user-defined object type.
typeof	Used to determine the type of a variable or expression
instanceof	Used to check if an object is an instance of a specific class or constructor
delete	Used to delete an object's property or element in an array
this	Refers to the current object. Its value depends on where it is used, such as in a method, function, or constructor.
class	Introduced in ES6, used for creating classes and supporting object-oriented programming.
super	Used to call methods of a parent class.
extends	Used to create a subclass in class-based inheritance.
Export,import	Used in modules to export and import functionality between files.
Async,await	Used in asynchronous programming for handling promises.
yield	Used in generator functions to pause and resume the execution.



7. CONCLUSION

From JavaScript you gain the ability to create dynamic and interactive websites, enhancing user experiences. Understanding JavaScript enables you to manipulate the Document Object Model (DOM), allowing you to dynamically update content, handle user input, and respond to events. JavaScript proficiency facilitates asynchronous programming, crucial for efficiently managing tasks like fetching data from servers without freezing the user interface. JavaScript opens the door to full-stack development. With frameworks like Node.js, you can use JavaScript on the server side, allowing for a seamless integration of server and client-side logic. This full-stack capability broadens your skill set and enables you to build end-to-end web applications. These tools simplify the development of complex user interfaces, making it easier to create scalable and maintainable applications.