JAVASCRIPT

JavaScript is the world's most popular programming language.

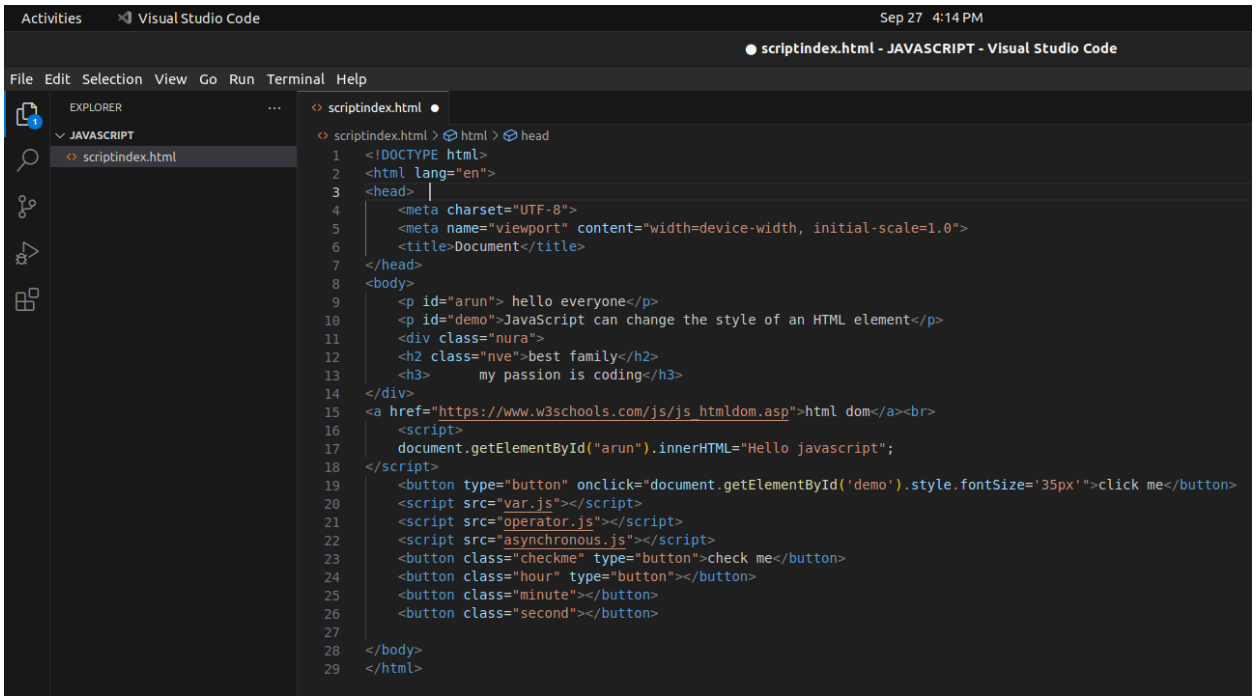
JavaScript is the programming language of the Web.

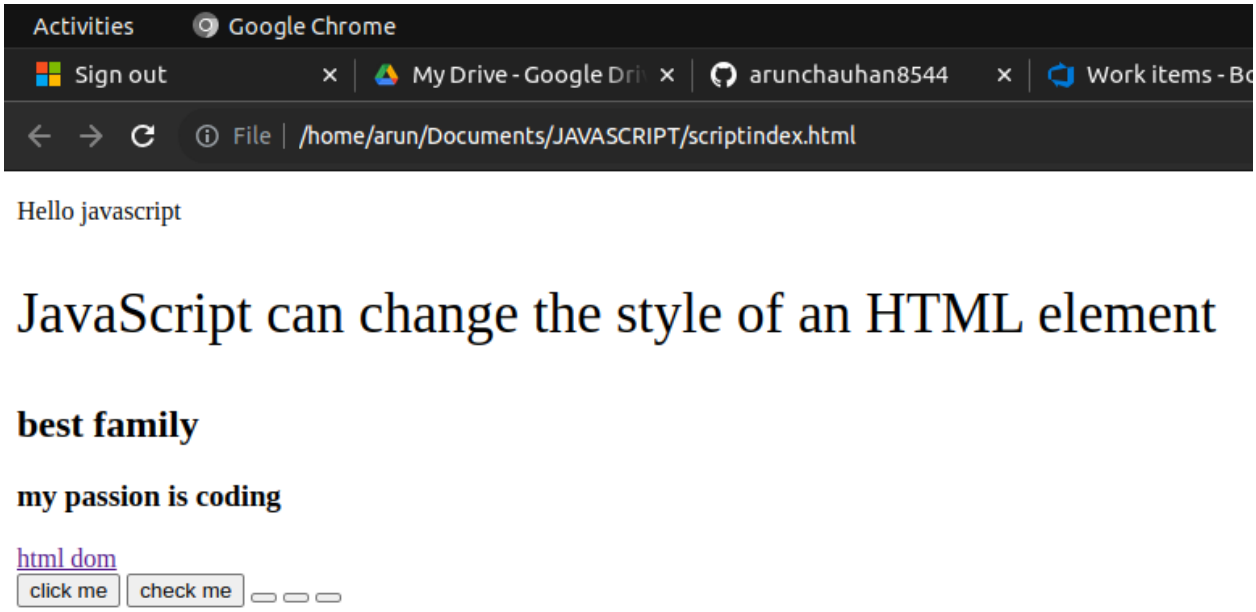
The <script> Tag

In HTML, JavaScript code is inserted between <script> and </script> Tags.

JavaScript Can Change HTML Content

One of many JavaScript HTML methods is getElementById().





External References

An external script can be referenced in 3 different ways:

- With a full URL (a full web address)
- With a file path (like /js/)
- Without any path

JavaScript Display Possibilities

JavaScript can "display" data in different ways:

- Writing into an HTML element, using innerHTML.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().
- Writing into the browser console, using console.log().

JavaScript Variables

JavaScript Variables can be declared in 4 ways:

- Automatically
- Using var
- Using let
- Using const

You cannot re-declare a variable declared with let or const.

jsvariable.js - JAVASCRIPT - Visual Studio Code

rminal Help

JS jsvariable.js × | JS operators.js | JS datatypes.js | JS ifelse.js | JS statswitch.js | JS functions.js | JS loop.js | JS object.js | JS string.js

JS jsvariable.js > ...

```javascript
var x=4;//x is a var variable
console.log(x);
let y=34;//y is a let variable
console.log(y);
const c=6.889;//c is a constant variable
console.log(c);
a=4;//automatically
console.log(a);
var carname;//creation of variable
console.log(carname);
carname="volvo"//assingn value to variable
let person="arun",gender="male";//one statement many variable
console.log(person);
let abhi;//undefined value
console.log(nura);
let $="hello world"//valid variable name
var  b = 10;
// Here b is 10

{
var b = 2;
// Here b is 2
}
// Here b is 2
console.log(b);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$ node jsvariable.js
4
34
6.889
4
undefined
arun
/home/arun/Documents/JAVASCRIPT/jsvariable.js:15
```

Types of JavaScript Operators

There are different types of JavaScript operators:

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- String Operators
- Logical Operators
- Bitwise Operators

- Ternary Operators
- Type Operators

operators.js - JAVASCRIPT - Visual Studio Code

rminal Help

jsvariable.js | operators.js | datatypes.js | ifelse.js | statswitch.js | functions.js | loop.js | object.js | string.js

operators.js > ...

```javascript
//arithmatic operator
let x=35;
let y=7;

console.log(x + y); // Addition
console.log(x- y); // Subtraction
console.log(x * y); // Multiplication
console.log(x / y); // Division
console.log(x % y); // Modulo

// assignment Operators
let c=10;//=operator
c+=30;
console.log(c);

//comparison operator
console.log(x == y);
console.log(x === y);
console.log(x!=y);
console.log(x !== y);
console.log(x + y);
//string operator
let text1 = "A";
let text2 = "B";
console.log(text1 <text2);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$ node operators.js
42
28
245
5
0
40
false
false
true
true
42
true
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$
```

JavaScript Data Types

JavaScript has 8 Datatypes

1. String
2. Number

3. Bigint
4. Boolean
5. Undefined
6. Null
7. Symbol
8. Object

The Object Datatype

The object data type can contain:

1. An object
2. An array
3. A date

Javascript numbers are always one type:
double (64-bit floating point).

The typeof Operator

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

An empty string has both a legal value and a type.

datatypes.js - JAVASCRIPT - Visual Studio Code

rminal Help

jsvariable.js | operators.js | datatypes.js | ifelse.js | statswitch.js | functions.js | loop.js | object.js | string.js

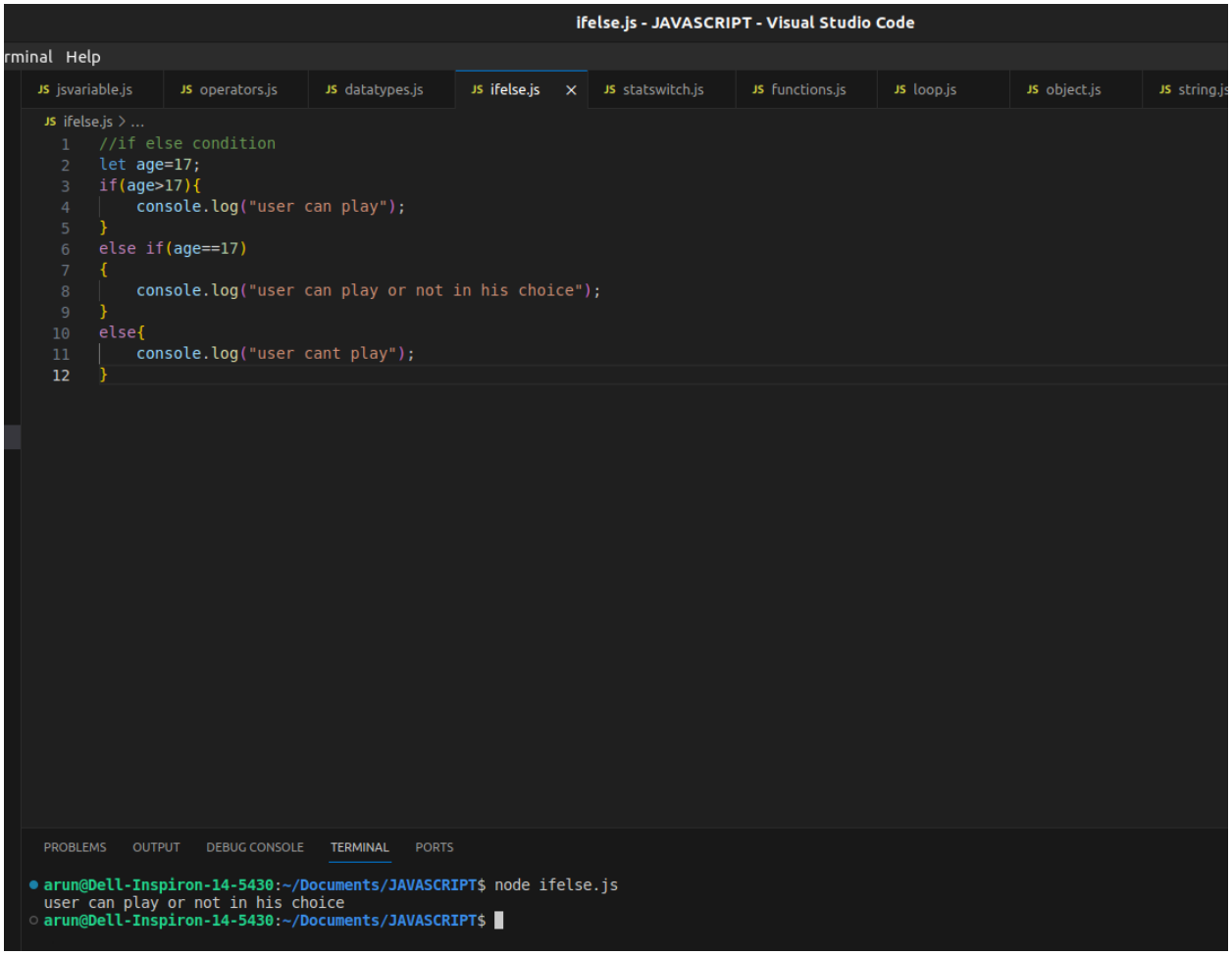
datatypes.js > ...

```javascript
//string datatype
let teacher="arun";
console.log(teacher);

//number datatype
let num=16;
console.log(num);

//boolean datatype
let x=true;
console.log(x);

let y;//undefined
console.log(y);

//object
const customer={firstname:"Arun",lastname:"chauhan"};
console.log(customer);

//array object
const color=["yellow","red","green"];
console.log(color);

//date object
const date=new Date("2022-03-25");
console.log(date);
let d=typeof (num);
console.log(d);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$ node datatypes.js
arun
16
true
undefined
{ firstname: 'Arun', lastname: 'chauhan' }
[ 'yellow', 'red', 'green' ]
2022-03-25T00:00:00.000Z
number
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$
```

JavaScript if, else, and else if



The JavaScript Switch Statement

statswitch.js - JAVASCRIPT - Visual Studio Code

rminal Help

JS jsvariable.js | JS operators.js | JS datatypes.js | JS ifelse.js | JS statswitch.js | JS functions.js | JS loop.js | JS object.js | JS string.js

JS statswitch.js > ...

```javascript
//conditional operator
let age=18;
let drink= age>=5? "coffee" : "milk";
console.log(drink);
let day=3;
switch(day){
    case 0:
        console.log("sunday");
        break;
    case 1:
        console.log("monday");
        break;
    case 2:
        console.log("sunday");
        break;
        case 3:
        console.log("sunday");
        break;
        case 4:
        console.log("sunday");
        break;
        case 5:
        console.log("sunday");
        break;
        case 6:
        console.log("sunday");
        break;
        default:
            console.log("not found");
            break;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$ node statswitch.js
coffee
sunday
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$
```

JavaScript Functions

A JavaScript function is a block of code designed to perform a particular task.A JavaScript function is executed when "something" invokes it (calls it).

functions.js - JAVASCRIPT - Visual Studio Code

rminal Help

jsvariable.js | operators.js | datatypes.js | ifelse.js | statswitch.js | functions.js | loop.js | object.js | string.js

functions.js > ...

```javascript
x1=6;
x2=8;
console.log(multiple(x1,x2));//function call
//function declaration
function multiple(p1,p2){
    return p1*p2;
}
const x= function (a,b) {return a*b};//anonymous function
console.log(x(3,4));
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$ node functions.js
48
12
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$
```

JavaScript Loop

Different Kinds of Loops

JavaScript supports different kinds of loops:

- for - loops through a block of code a number of times
- for/in - loops through the properties of an object

- for/of - loops through the values of an iterable object
- while - loops through a block of code while a specified condition is true

● do/while - also loops through a block of code while a specified condition is true

loop.js - JAVASCRIPT - Visual Studio Code

rminal Help

jsvariable.js | operators.js | datatypes.js | ifelse.js | statswitch.js | functions.js | loop.js | object.js | string.js

loop.js > car > model

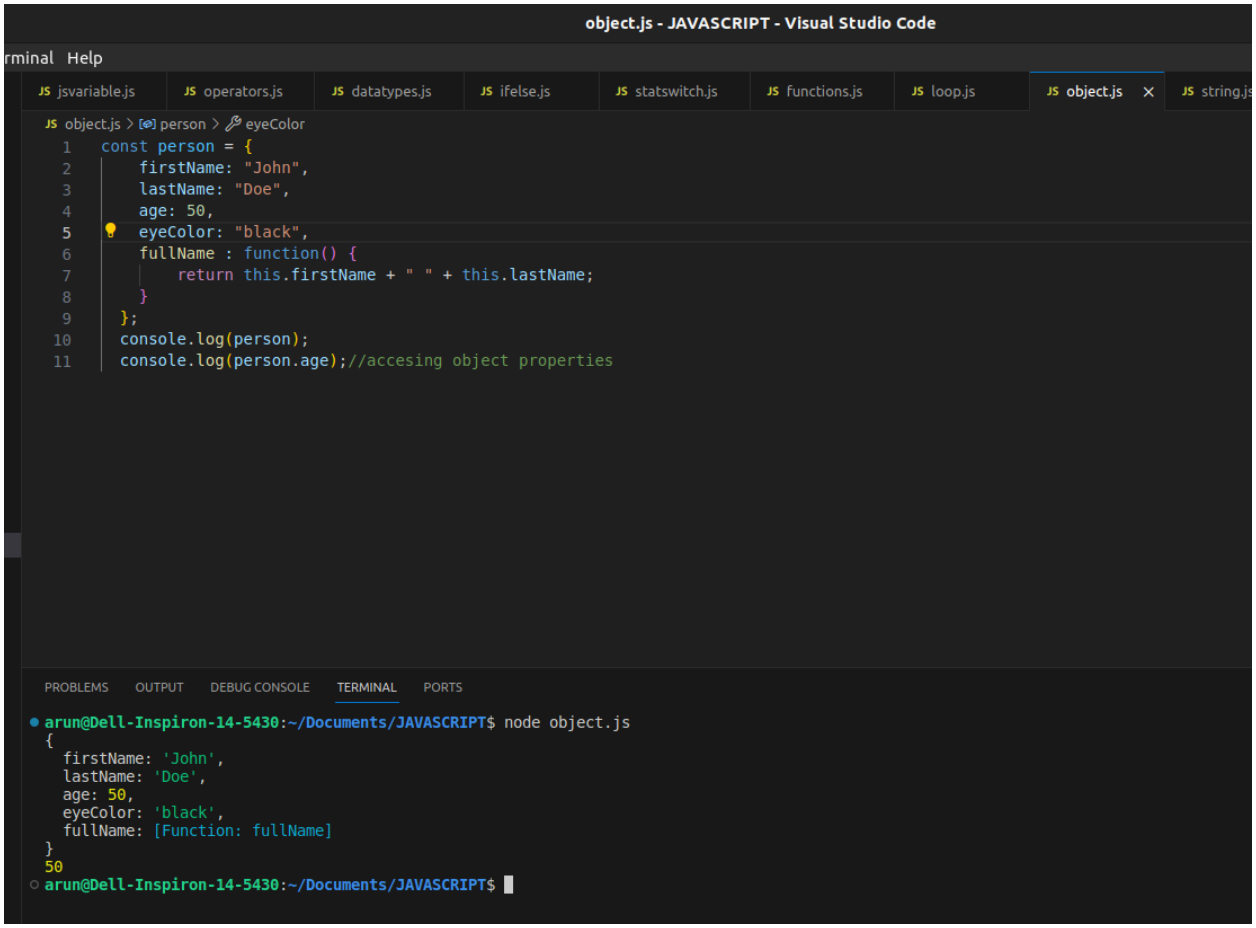
```javascript
//while loop
let i=10;
while(i<15)
{
    console.log(i);
    i++;
}
//for loop
for(let j=0;j<20;j++)
{
    console.log(j);
}
//do while loop
let k=10;
do
{
    console.log(k);
    k++;
}while(k<9);
//for loop in array
let fruits=["apple","mango","grapes","banana"];
for(let i=0;i<fruits.length;i++)
{
    console.log(fruits[i]);
}
//for in loop in object
let car={
    model:2020,
    color:"black",
    company: "toyoto"
}
console.log(car);
for(let key in car)
{
    console.log(car[key]);
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
16
17
18
19
10
apple
mango
grapes
banana
{ model: 2020, color: 'black', company: 'toyoto' }
2020
black
toyoto
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$
```

JavaScript Objects

The values are written as name:value pairs (name and value separated by a colon).



JavaScript Strings

JavaScript strings are for storing and manipulating text.

Javascript string methods-

1. **Length**: The length property returns the number of characters in the string. It doesn't require parentheses as it is not a method but a property of the string.

2. **Character Access**: Strings are zero-indexed, meaning you can access individual characters using their index positions. You can use the charAt(index) method or directly access the characters using square brackets notation (e.g., str[0]).

3. **Concatenation**: The concat() method is used to join two or more strings together, creating a new string that includes all the concatenated parts.

4. **Substring Extraction**: You can extract a substring from a string using methods like substring(startIndex, endIndex) and slice(startIndex, endIndex). Both methods return the extracted substring based on the provided indices. substring() will swap the indices if startIndex is greater than endIndex, while slice() allows negative indices to count from the end of the string.

5. **Case Conversion**: To convert the case of a string, you can use toLowerCase() and toUpperCase() methods, which return new strings with all characters in either lowercase or uppercase, Respectively.

6**. Trimming**: The trim() method removes leading and trailing whitespaces from a string, creating a new string with the trimmed Content.

7. **Searching:** For finding substrings within a string, you can use indexOf(substring, startIndex) method. It returns the index of the first occurrence of the substring or -1 if the substring is not found. To search from the end of the string, you can use lastIndexOf(substring, startIndex).

8. **Splitting:** The split(separator) method is used to split a string into an array of substrings based on the specified separator. For example, splitting a comma-separated string into an array of Values.

9. **Replacing**: The replace(searchValue, replaceValue) method

string.js - JAVASCRIPT - Visual Studio Code

rminal Help

JS jsvariable.js | JS operators.js | JS datatypes.js | JS ifelse.js | JS statswitch.js | JS functions.js | JS loop.js | JS object.js | JS string.js

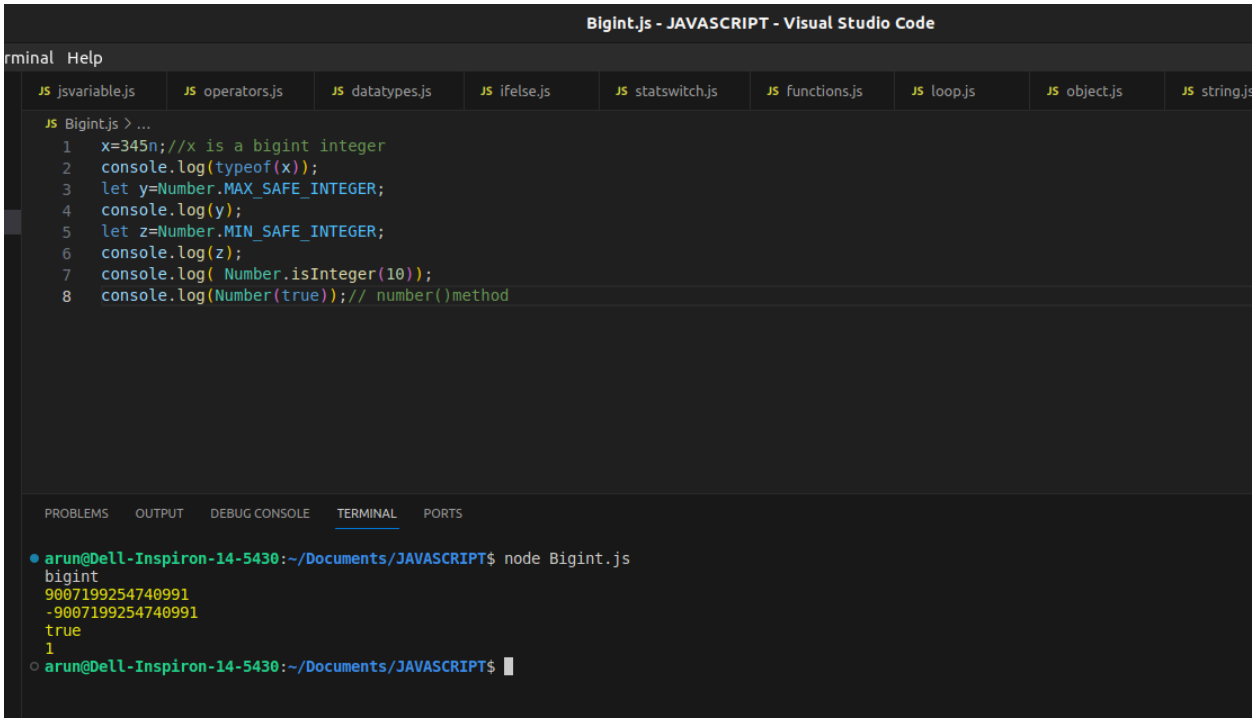
JS string.js > arun

```javascript
let firstName="Arun";
console.log(firstName[1]);//string indexing
let lastName="   Chauhan   "
console.log(lastName);
let newString=lastName.trim();//trim()
console.log(lastName.length);//length()
console.log(newString.length);
let text="abcdEfgHijkLMn";
console.log(text.toUpperCase());//toUpperCase()
console.log(text.slice(4,8));//string slice()
//convert string to a number
let num=+"abcdEfgHijkLMn";
console.log(typeof(num));
console.log(typeof(text));
//convert number to string
num=num+""
console.log(typeof(num));
num=Number(num);
console.log(typeof(num));
num=String(num);
console.log(typeof(num));
let text2="welome to js";
let newtext=text2.replace("js","css");
text3= "Hello" + " " + "World!";//string concatination
console.log(text3);
newarr1=text3.split(" ");
console.log(newarr1[1]);
let arun=`arun chauhan`;//template literals
console.log(arun);
let text4= `Welcome ${text2}, ${arun}!`;
console.log(text4);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$ node string.js
r
   Chauhan
13
7
ABCDEFGHIJKLMN
EfgH
number
string
string
number
string
Hello World!
World!
arun chauhan
Welcome welome to js, arun chauhan!
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$
```

JavaScript BigInt



JavaScript Arrays

An array is a special variable, which can hold more than one value:

Arrays.js - JAVASCRIPT - Visual Studio Code

rminal Help

JS jsvariable.js | JS operators.js | JS datatypes.js | JS ifelse.js | JS statswitch.js | JS functions.js | JS loop.js | JS object.js | JS string.js

JS Arrays.js > vars

```javascript
const names=["arun","nura","tarun"];//creation of an array
console.log(names[0]);
const vars=new Array("saab","volvo","bmw");
console.log(vars);
console.log(names.toString());//convert array to string
console.log(names);
names.push("ashish");//add element
console.log(names);
console.log(names.length);// js array length
console.log(names.pop());
//array concatenation
const myGirls = ["Cecilie", "Lone"];
const myBoys = ["Emil", "Tobias", "Linus"];
const myChildren = myGirls.concat(myBoys);
console.log(myChildren);
names.splice(2,0,"shreya","dk");//splice method
console.log(names);
const fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
const citrus = fruits.slice(1);//slice method
console.log(citrus);
console.log(names.sort());//sort an array
console.log(names.reverse());//reverse an array
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$ node Arrays.js
arun
[ 'saab', 'volvo', 'bmw' ]
arun,nura,tarun
[ 'arun', 'nura', 'tarun' ]
[ 'arun', 'nura', 'tarun', 'ashish' ]
4
ashish
[ 'Cecilie', 'Lone', 'Emil', 'Tobias', 'Linus' ]
[ 'arun', 'nura', 'shreya', 'dk', 'tarun' ]
[ 'Orange', 'Lemon', 'Apple', 'Mango' ]
[ 'arun', 'dk', 'nura', 'shreya', 'tarun' ]
[ 'tarun', 'shreya', 'nura', 'dk', 'arun' ]
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$
```

JAVASCRIPT DATE METHOD-

JavaScript's Date object provides methods to work with dates and times.

Here's a summary of the theory behind some commonly used date methods without specific syntax:

1. **new Date():** Creates a new Date object representing the current date and time.

2. **Date.parse(dateString)**: Parses a date string and returns the

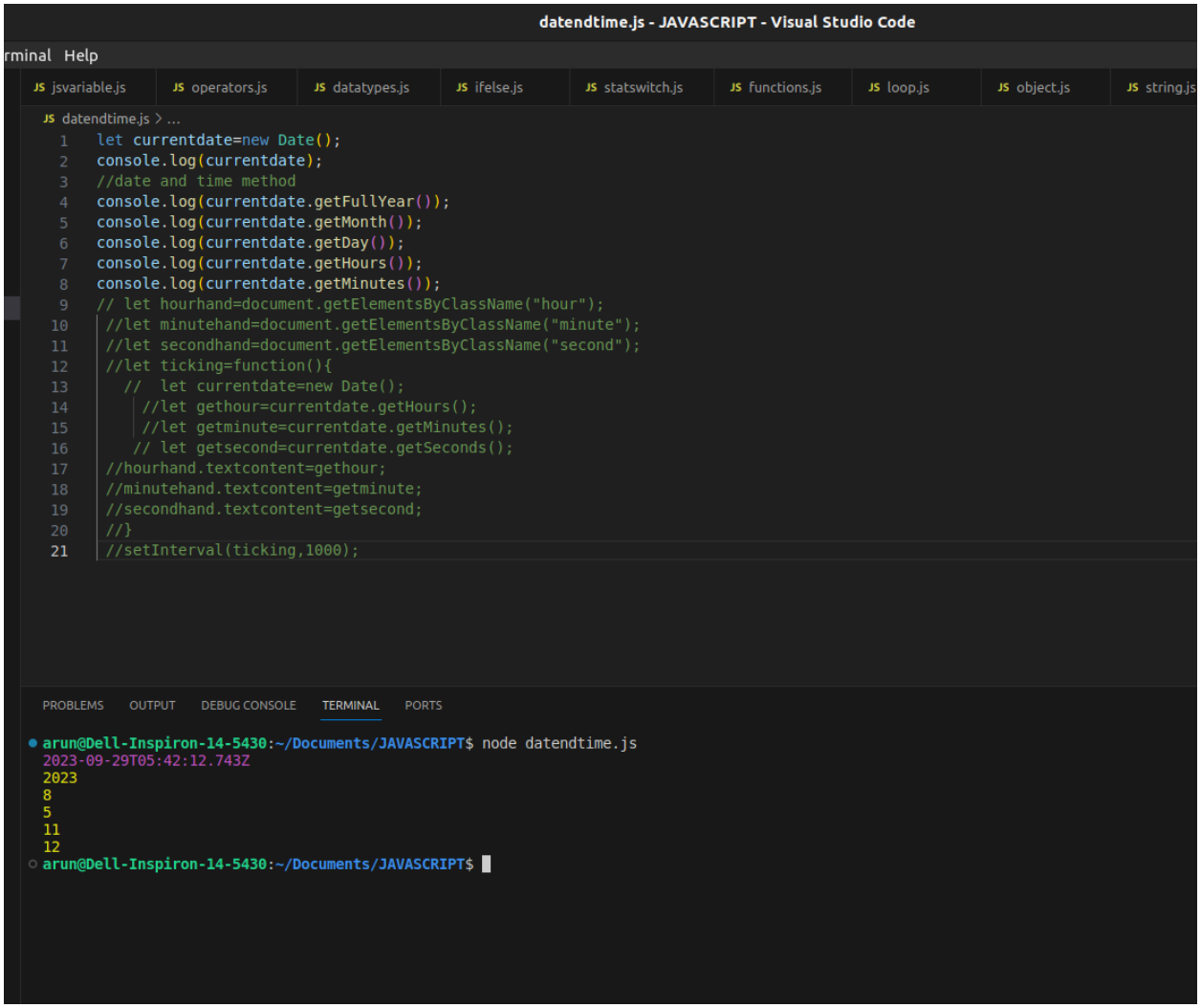
number of milliseconds since January 1, 1970 (Unix Epoch).
3. **getFullYear(**): Gets the year (4-digit) of the Date object.

4. **getMonth():** Gets the month (0-indexed) of the Date object (0 for January, 1 for February, and so on).

5. **getDate()**: Gets the day of the month (1-31) of the Date object.

6. **getDay(**): Gets the day of the week (0-6) of the Date object (0 for Sunday, 1 for Monday, and so on).

7. getHours(), getMinutes(), getSeconds(), getMilliseconds(): Gets the hours (0-23), minutes (0-59), seconds (0-59), and milliseconds (0-999) of the Date object.



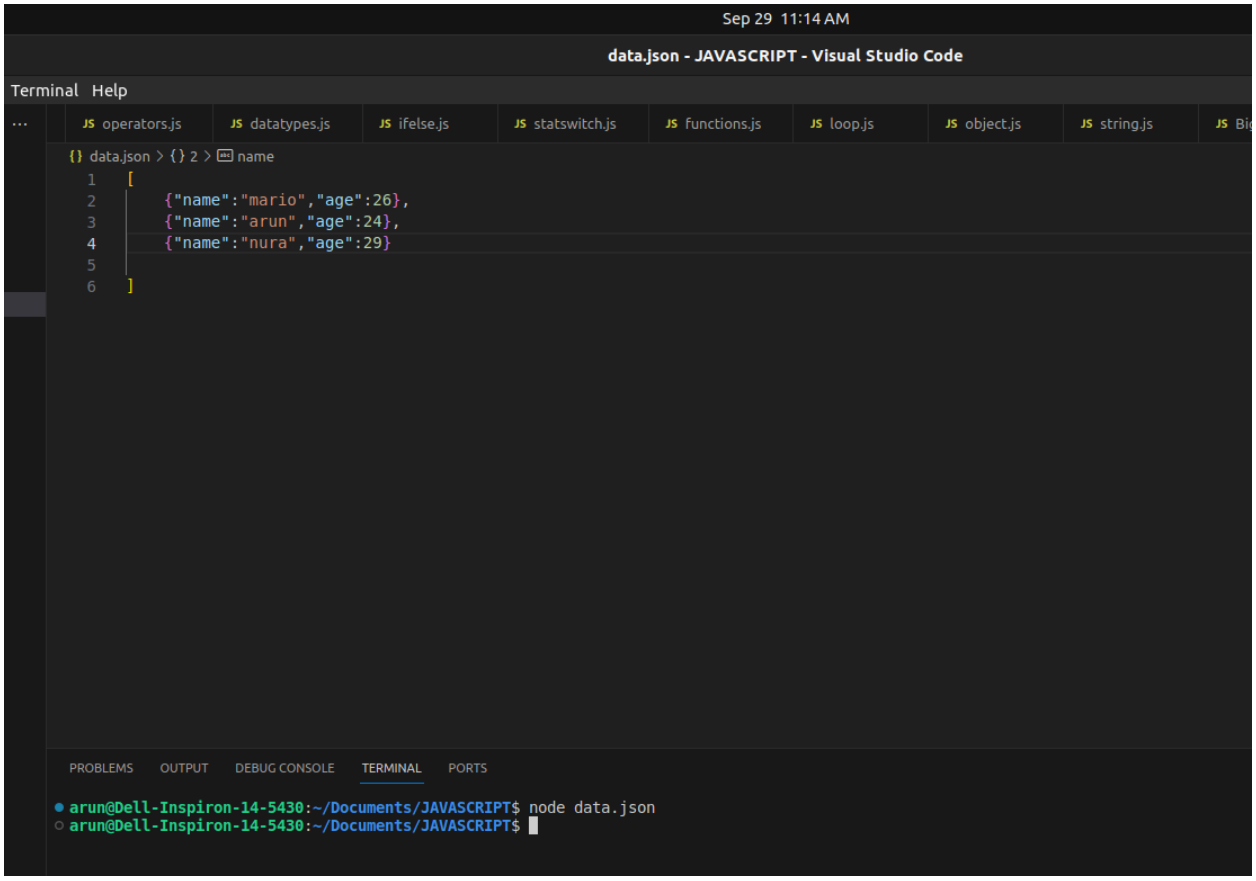
JavaScript Hoisting

Hoisting is JavaScript's default behavior of moving declarations to the top.

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.



JAVASCRIPT DOM

Sep 29 11:15 AM

dom.js - JAVASCRIPT - Visual Studio Code

Terminal Help

datatypes.js | ifelse.js | statswitch.js | functions.js | loop.js | object.js | string.js | Bigint.js | Ar

dom.js > parentElement

```javascript
//Javascript DOM(document object model)
//created by browseras the html load into the browser
//browser create object of html called "document object"
//in document object there is model of complete html in tree like structure
//console.log(document.URL);

//getting html elements using querySelector/All
//queryselector returns "first element" that matches css selector
//to get all matched selector we use querySelectorAll

let ans=document.querySelector('p');
console.log(ans);
let ans2=document.querySelectorAll('p');//select all gives nodelist type
console.log(ans2);
let ans3=document.querySelector('#arun');
console.log(ans3);
//getElementsByTagName
let tagname=document.getElementsByTagName('p');
console.log(tagname);//this gives html collection type
//foreach method weill not work on collection.
//getElemntByClassName you can also use  this.
//getElemntById
let id=document.getElementById('arun');
console.log(id);
//updating and changing the content
//inner text,ignore spaces
//retrieve and set content in plain text
let heading=document.querySelector('.nve');
console.log(heading.innerText);

//innertext,it does not ignore spaces
//retrieve and set content in html formate
console.log(heading.innerHTML);
heading.innerText +="arunchauhan";//update
//if u dont want to delete previous text then add +
console.log(heading.innerText);
//getting and setting attribute of element
let htmldomlink=document.querySelector('a');
console.log(htmldomlink.getAttribute('href'));
```

JS datatypes.js | JS ifelse.js | JS statswitch.js | JS functions.js | JS loop.js | JS object.js | JS string.js | JS Bigint.js | JS Ar

JS dom.js > parentElement

```javascript
console.log(htmldomlink.getAttribute('href'));
//set the attribute
htmldomlink.setAttribute('href',"https://www.w3schools.com/css/default.asp");
console.log(htmldomlink.getAttribute('href'));
htmldomlink.innerText="css tutorial";
console.log(htmldomlink.innerText);
//adding the style
let color1=document.querySelector('h2');
color1.style.color="red";
color1.style.backgroundColor="yellow";

//add,remove and replace class of element
//this also helps to change css dynamically
let heading2=document.querySelector('h2');
heading.classList.add('newClass');
//remove the class
heading.classList.remove("newClass");
//replace the class
heading.classList.replace("nve","newClass");
//parent,children and sibiling elements
let parentElement=document.querySelector('.nura');
//all children of parent
console.log(parentElement.children);
//we can not run foreach method on html collection so
//first convert it into array
console.log(Array.from(parentElement.children));
Array.from(parentElement.children).forEach(function(element)
{
element.classList.add("coders");
})
let childElement=document.querySelector('.nve');

//console.log(childElement.parentElement);
//find next sibiling of child
//console.log(childElement.nextElementSibling);
//console.log(childElement.previousElementSibling);
//event basics(click event)
let eventElement=document.getElementsByClassName('checkme');
console.log(eventElement);
//addeventlistner
// eventElement.addEventListener("click", function(){
// console.log("clicked me")
// })
//form events(submit form)
```

Activities | Google Chrome | Sep 29 11:19 AM

Sign out | My Drive - Google Drive | arunchauhan8544 | Work items - Boards | JAVASCRIPT/pro

File | /home/arun/Documents/JAVASCRIPT/scriptindex.html

Hello javascript

JavaScript can change the style of an HTML element

best family

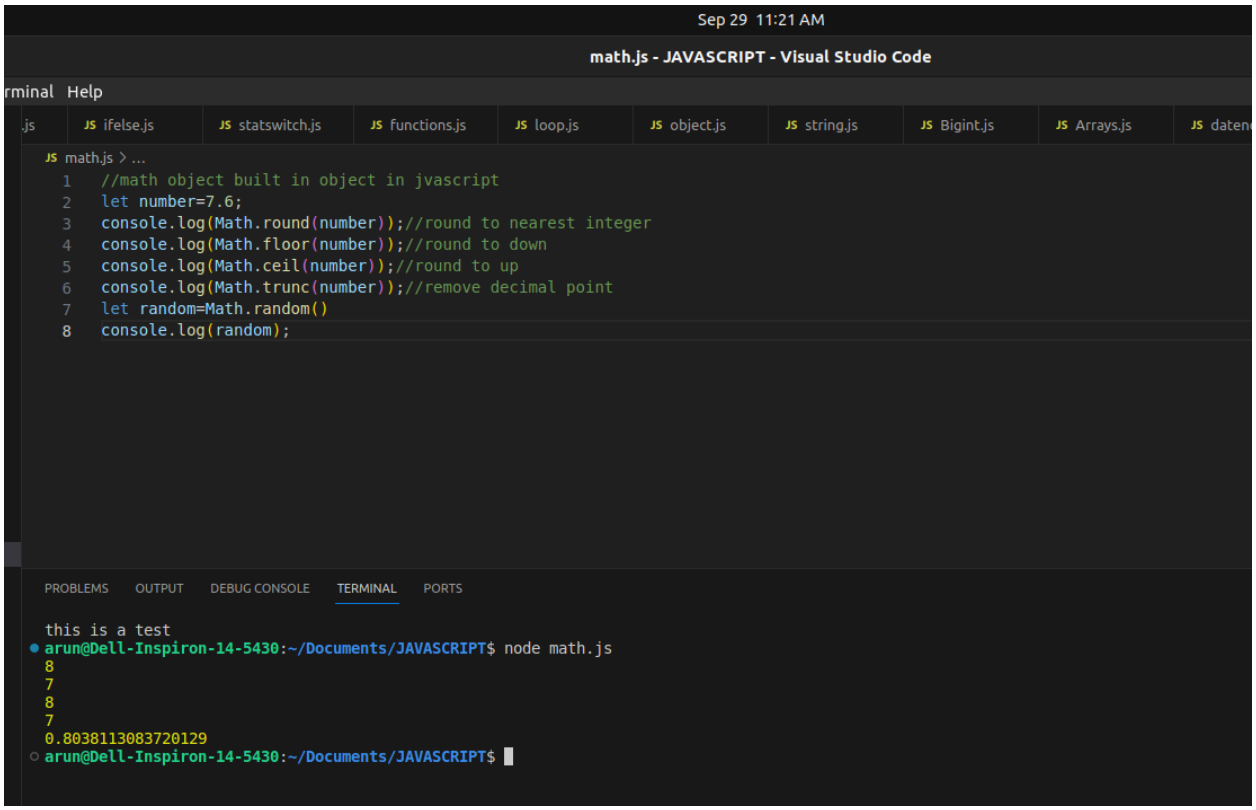
my passion is coding

html dom

click me | check me

JavaScript Math Object

The JavaScript Math object allows you to perform mathematical tasks on Numbers.



Call and apply method-

Sep 29 11:22 AM

function_call_apply.js - JAVASCRIPT - Visual Studio Code

rminal Help

statswitch.js | functions.js | loop.js | object.js | string.js | Bigint.js | Arrays.js | datendtime.js | data.json

function_call_apply.js > user > lastname

```javascript
//withbthe help of call and apply we can manually change the value
//of this keyword
let details={
    firstname: "arun",
    lastname:"chauhan",
    fullname:function(){

        console.log(this.firstname +" "+this.lastname);
    }
}
let person1={
    firstname: "shrey",
    lastname:"gupta",
   age:24

}
let newdetails=details.fullname;
newdetails.call(person1);
//apply method
//there is a difference in call and apply in arguments
//in apply we have to pass arguments in an array
const person = {
    fullName: function(city, country) {
      console.log(this.firstName + " " + this.lastName + "," + city + "," + country);
    }
  }

  const person2= {
    firstName:"John",
    lastName: "Doe"
  }

  person.fullName.apply(person2, ["Oslo", "Norway"]);
  //bind method
function greet()
{
    console.log(this.firstname + " "+this.lastname);
}
let user={
    firstname: "arun",
    lastname: "chauhan"
}
let greets= greet.bind(user);
greets()
```

```
}
let newdetails=details.fullname;
newdetails.call(person1);
//apply method
//there is a difference in call and apply in arguments
//in apply we have to pass arguments in an array
const person = {
    fullName: function(city, country) {
      console.log(this.firstName + " " + this.lastName + "," + city + "," + country);
    }
  }

  const person2= {
    firstName:"John",
    lastName: "Doe"
  }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$ node function_call_apply.js
shrey gupta
John Doe,Oslo,Norway
arun chauhan
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$
```

Foreach method

Sep 29 11:25 AM

foreach.js - JAVASCRIPT - Visual Studio Code

erminal Help

functions.js | loop.js | object.js | string.js | Bigint.js | Arrays.js | datendtime.js | data.json | dom.js

foreach.js > myfunction

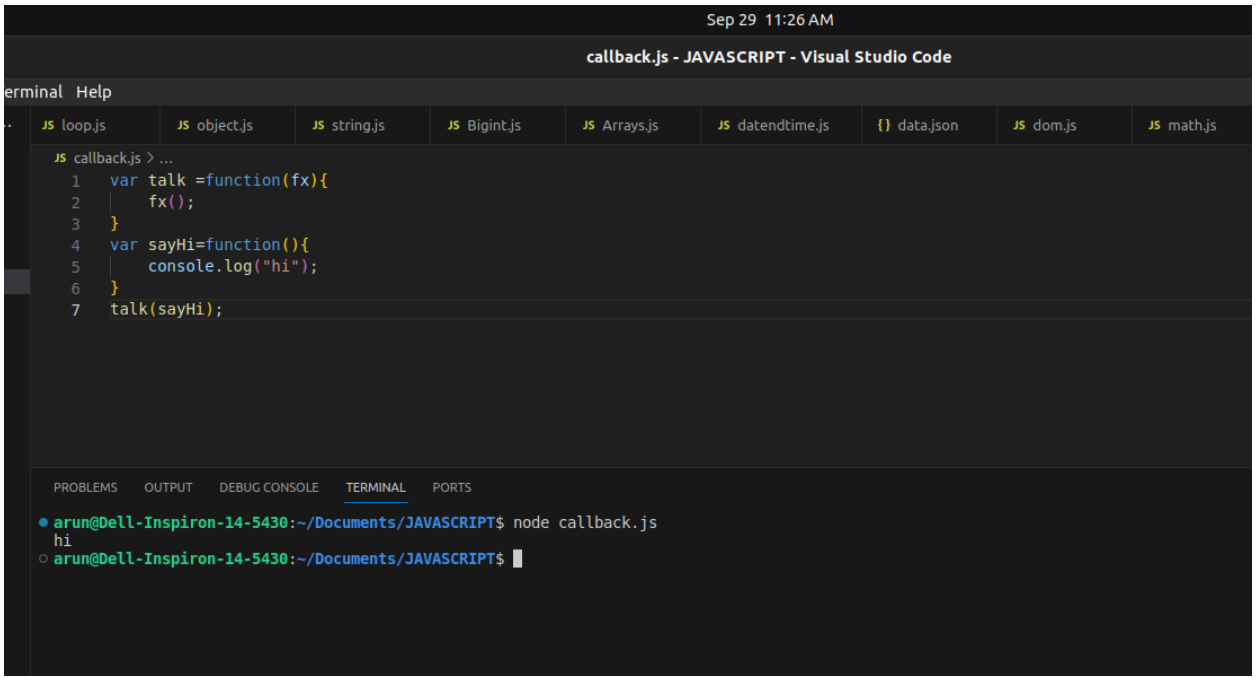
```
let numbers=[45,5,9,67,987];
let txt="";
numbers.forEach(myfunction);
function myfunction(x)
{
    txt=x;
    console.log(txt);
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$ node foreach.js
45
5
9
67
987
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$
```

JAVASCRIPT CALLBACK FUNCTION-

In JavaScript, a callback function is a function that is passed as an argument to another function and is intended to be executed at a later time or after the completion of an asynchronous operation. Callback functions are a fundamental concept in JavaScript, especially when dealing with asynchronous code, event handling, and functional programming.



Asynchronous and and AJAX

Async/Await:

"async" and "await" are keywords used in JavaScript to simplify working with asynchronous code. "async" is used to define asynchronous functions, while "await" is used inside an asynchronous function to pause its execution until a Promise is resolved or rejected. This makes the code look more synchronous and easier to read.

AJAX (Asynchronous JavaScript and XML):

AJAX is a technique in JavaScript that allows web pages to make asynchronous requests to the server without reloading the entire page. It enables data exchange between the web browser and the server in the background, providing a more responsive and interactive user experience. AJAX requests can retrieve data from the server, send data to the server, or interact with web APIs.

Sep 29 11:28 AM

asyncndazax.js - JAVASCRIPT - Visual Studio Code

erminal Help

object.js | string.js | Bigint.js | Arrays.js | datendtime.js | data.json | dom.js | math.js | function_call_app

asyncndazax.js > ...

```javascript
//asynchronous javascript
//async code example

console.log(1);
console.log(2);
console.log(3);

//async code
setTimeout(()=>{
    console.log("print after 5 sec");
},5000)//after 5 sec
console.log(3);
console.log(4);
//azax code//asynchronous example
//making http request(example)
let todos=(callback)=>{
let request= new XMLHttpRequest();
console.log(request);
request.addEventListener('readystatechange',()=>{
    if(request.readystate==4)
       {
        console.log(undefined,request.responseText);
       }
       else{
        console.log("error comes",undefined);
       }
})
//set up to request
request.open("Get","https://jsonplaceholder.typicode.com/todos")


//sending the request
request.send()}

console.log(6);
todos((error,data)=>
{
    if(error)
    {
        console.log(error);
    }
    else{
        console.log(data);
    }
})
console.log(7);
console.log(8);
```

```
19  request.addEventListener("readystatechange",()=>{
20      if(request.readystate==4)
21          {
22           console.log(undefined,request.responseText);
23          }
24          else{
25           console.log("error comes",undefined);
26          }
27  })
28  //set up to request
29  request.open("Get","https://jsonplaceholder.typicode.com/todos")
30
31
32  //sending the request
33  request.send()}
34
35  console.log(6);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
this is a test
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$ node asyncndazax.js
1
2
3
3
4
6
/home/arun/Documents/JAVASCRIPT/asyncndazax.js:17
let request= new XMLHttpRequest();
             ^

ReferenceError: XMLHttpRequest is not defined
    at todos (/home/arun/Documents/JAVASCRIPT/asyncndazax.js:17:14)
    at Object.<anonymous> (/home/arun/Documents/JAVASCRIPT/asyncndazax.js:36:1)
    at Module._compile (internal/modules/cjs/loader.js:999:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1027:10)
    at Module.load (internal/modules/cjs/loader.js:863:32)
    at Function.Module._load (internal/modules/cjs/loader.js:708:14)
    at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:60:12)
    at internal/main/run_main_module.js:17:47
arun@Dell-Inspiron-14-5430:~/Documents/JAVASCRIPT$
```

CALLBACK HELL-Callback hell, also known as "Pyramid of Doom," is a term used to describe a situation in JavaScript code where multiple nested callback functions make the code structure difficult to read and maintain.

Sep 29 11:32 AM

azaxndjson.js - JAVASCRIPT - Visual Studio Code

inal Help

ng.js | Bigint.js | Arrays.js | datendtime.js | data.json | dom.js | math.js | function_call_apply.js | foreach.js

azaxndjson.js > ...

```javascript
//callbackhell
let todos=(resourse,callback)=>{
    let request= new XMLHttpRequest();
    console.log(request);
    request.addEventListener('readystatechange',()=>{
        if(request.readyState==4)
           {
            callback(undefined,request.responseText);
           }
           else{
            callback("error comes",undefined);
           }
    })
    //set up to request
    request.open("Get",resourse)


    //sending the request
    request.send()}

    console.log(6);
    todos("data.json", (error,data)=>
    {
        if(error)
        {
            console.log(error);
        }
        else{
            console.log(data);
        }

    })

    console.log(8);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Promises

Promises help avoid the callback hell, a situation where nested callbacks make the code difficult to read and maintain.

A Promise can be in one of three states:

1. **Pending**: The initial state of the Promise, representing an ongoing asynchronous operation.

2. **Fulfilled/Resolved:** The Promise has successfully completed, and

the associated value (result) is available. This is when the resolve() function is called.

3. **Rejected:** The Promise encountered an error during execution. This is when the reject() function is called.