CSC-336 Web Technologies

Lecture 7-8

Topics:

- History of JavaScript, Console Programming,
- Data Types, Variables, String Operations, Arithmetic
- Functions, If else structure, loops

Muhammad Naveed Shaikh
Department of Computer Engineering
naveedshaikh@cuiatd.edu.pk

COMSATS University Islamabad | Abbottabad Campus

Introduction

- JavaScript is :
  - A high-level, interpreted programming language.
- It enables dynamic behavior on web pages.
- Plays a key role in:
  - front-end, back-end, and full-stack development.
- Alongside HTML & CSS, it forms the core of the web.

Early Beginnings (1995)

- Brendan Eich created JavaScript in 1995 while working at Netscape.
- The first internal name of JavaScript was Mocha,
  - chosen by Netscape’s management.
- It was later renamed LiveScript
  - when integrated into Netscape Navigator 2.0
- and finally JavaScript
  - after partnership with Sun Microsystems (creators of Java).

⚠️ Note: This “Mocha” is unrelated to the modern Mocha testing framework used in Node.js today.

Why “JavaScript”?

- Named “JavaScript” for marketing reasons.
- Aimed to ride the popularity of Java at that time.
- But, Java and JavaScript are completely different languages!

Feature	Java	JavaScript
Type	Object-oriented, compiled language	interpreted language
Object-based, Platform	Runs on JVM (Java Virtual Machine)	Runs in browsers and on Node.js
Syntax	Strict, strongly typed	Flexible, loosely typed
Execution	Needs compilation (.class files)	Runs directly in browser or runtime
Usage	Desktop, mobile (Android), backend apps	Web development, servers, mobile & AI
Example	System.out.println("Hello");	console.log("Hello");

Standardization (ECMAScript)

- In 1997, JavaScript was submitted to ECMA International.
- Standardized as ECMAScript (ES).
- Ensured compatibility across different browsers.
- ES1 (1997) was the first official standard.

Evolution of ECMAScript

- ES3 (1999): Regular expressions, better string handling
- ES5 (2009): JSON support, strict mode
- ES6 (2015): Classes, arrow functions, modules, promises
- ES7–ES13 (2016–2022): Async/await, optional chaining, modern syntax

Browser Wars & Growth

- 1990s: Browser competition between Netscape and Microsoft.
- Microsoft introduced JScript (their version of JS) in Internet Explorer.
- Led to incompatibilities and cross-browser issues.
- Pushed the need for standardization.

Rise of Modern JavaScript

- 2009: Node.js introduced JavaScript to the server-side.
- 2010s: Explosion of JS frameworks and libraries such as jQuery, AngularJS, React, Vue.js.
- JavaScript became one of the most popular programming languages in the world.

Today’s JavaScript Ecosystem

- Front-end: React, Vue, Angular
- Back-end: Node.js, Express.js
- Mobile: React Native, Ionic
- Desktop: Electron
- AI & ML: TensorFlow.js
- Truly a universal language.

Timeline Summary

- 1995: Birth of JavaScript (Netscape)
- 1997: ECMAScript standardization
- 2009: Node.js introduced
- 2015: ES6 brings modern features
- 2020s: Dominates full-stack, mobile & AI development

Fun Facts

- JavaScript was created in 10 days.
- Runs in all modern browsers without installation.
- Powers over 98% of websites today.
- Its mascot is often a yellow JS logo ⚡

Conclusion: History & Intro of JavaScript

- JavaScript evolved from a simple scripting tool to a powerful ecosystem.
- It’s now the backbone of interactive web applications.
- Continuous updates make it future-ready.
- JavaScript: the language that never stops evolving!

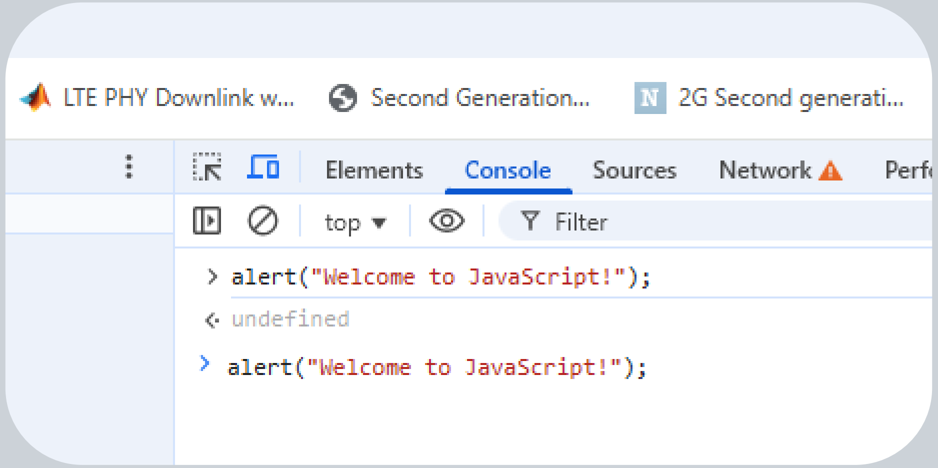
Adding Behavior to a Webpage Using JavaScript

- JavaScript adds behavior and interactivity to webpages.

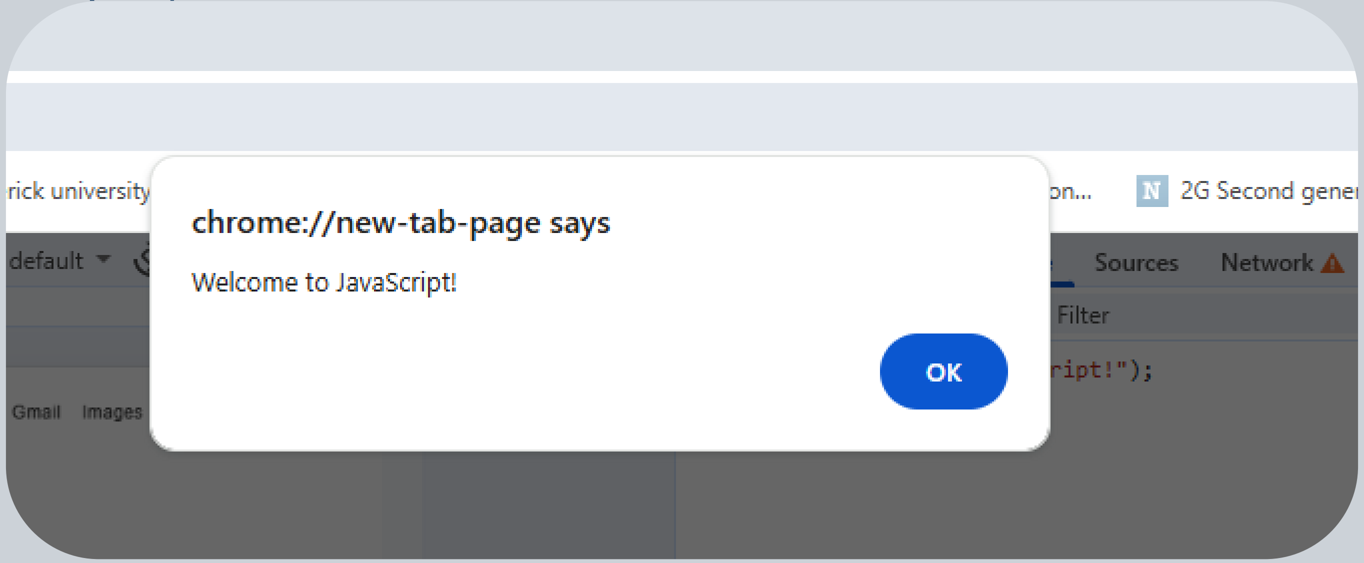
1. Open Google Chrome
2. Right-click → Inspect
3. Go to the Console tab
4. Type JavaScript directly and see instant results!

Type each line in the Chrome Console and press Enter:

- alert("Welcome to JavaScript!");



```
> alert("Welcome to JavaScript!");
<· undefined
> alert("Welcome to JavaScript!");
```



chrome://new-tab-page says

Welcome to JavaScript!

What’s Happening Here

- • The browser reads your JavaScript command
- • alert() tells the browser: “Show this message.”
- • No HTML change needed — you directly interacted with the page!

What Are Data Types?

- Data type = kind of information a variable holds
- Examples in real life:
  - Your name → text
  - Your age → number
  - Is student? → true/false
- In JavaScript, everything has a type.

Primitive Data Types

The 7 main primitive types in JavaScript:

1. String ➔text
2. Number ➔numeric values
3. Boolean ➔ true or false
4. Undefined➔ not assigned yet
5. Null ➔intentionally empty
6. Symbol ➔unique identifier (advanced)
7. BigInt ➔large integers (advanced)

Strings (Text)

- Used for words, sentences, or any characters.

- Examples:
  - "Hello"
  - 'JavaScript'
  - "123"   // still text
- Try in console:
  - alert("I am learning JavaScript!");
  - typeof "Hello";

Numbers

- Used for math or numeric data.
- Examples:
  - 5
  - 3.14
  - -10
- Try in console:
  - alert(2 + 3);
  - typeof 2.5;

Booleans

- Represent true or false values — helpful in decisions.
- Examples:
  - true
  - false
- Try in console:
  - alert(true);
  - alert(5 > 2);
  - typeof false;

Undefined and Null

- • Undefined: variable declared but not given a value
- • Null: value purposely set to “nothing”

- Examples:
  - let a;
  - alert(a); // undefined

  - let b = null;
  - alert(b); // null

typeof Operator

- Used to check what kind of data you have.

- Try in console:
  - typeof "COMSATS"
  - typeof 123
  - typeof true
  - typeof null

- Note: typeof null returns "object" (a known JavaScript quirk).

Quick Summary

- .

Type	Example	Description
String	"Hello"	Text values
Number	42, 3.14	Numeric values
Boolean	true, false	Logic values
Undefined	let x;	No value assigned
Null	let y = null;	Intentionally empty value

Javascript Naming Conventions for variable

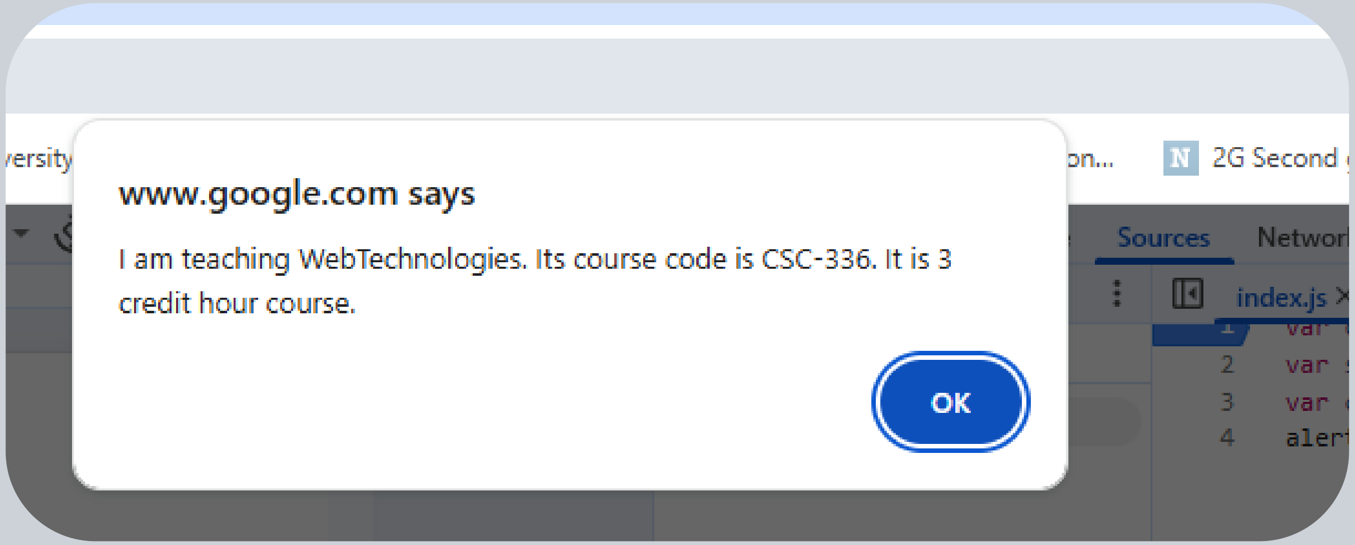
- .

Rule / Tip	Example	Explanation
✅ **Use camelCase**	let userName = "Ali";	Start with lowercase, capitalize new words. Common in JS.
✅ **Start with a letter, _, or $**	let _score = 10;, let $price = 99;	Variable names **cannot** start with numbers.
❌ **Don’t use spaces or special characters**	❌ let user name = "Ali";	Use camelCase instead: userName.
✅ **Be descriptive**	let totalMarks = 500;	Names should tell what the variable stores.
❌ **Avoid JS reserved words**	❌ let for = 5;	Words like for, if, var, function cannot be variable names.
⚠️ **Case-sensitive**	userName ≠ username	JavaScript treats them as different variables.

String Concatenation

- String concatenation means **joining two or more strings together** to make a single string.

```
var creditHours=3;
var subName="Web Technologies";
var courseCode="CSC-336";
alert("I am teaching"+subName+". Its course code is "+courseCode+". It is "+creditHours+" credit hour course.");
```

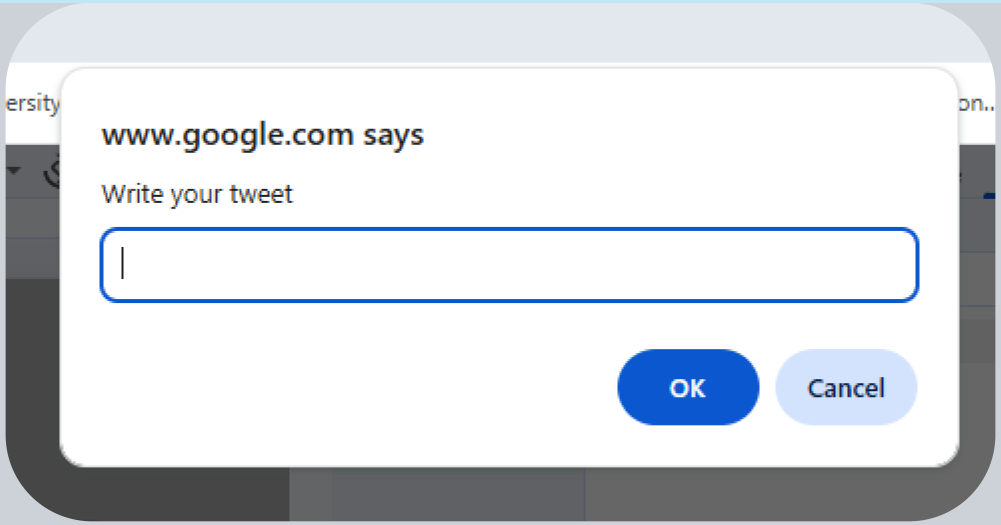


www.google.com says

I am teaching WebTechnologies. Its course code is CSC-336. It is 3 credit hour course.

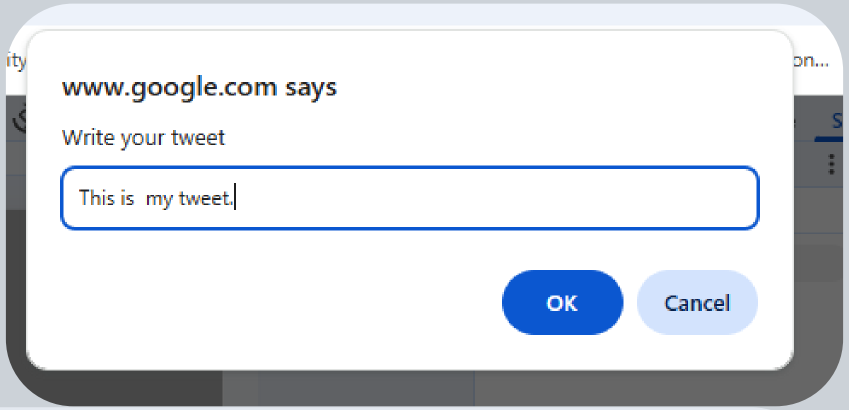
Tweet Web App

```
var tweet = prompt ("Write your tweet");
alert ("your tweet has "+tweet.length+" characters. your remaining characters are "+(240-tweet.length)+".");
```



www.google.com says

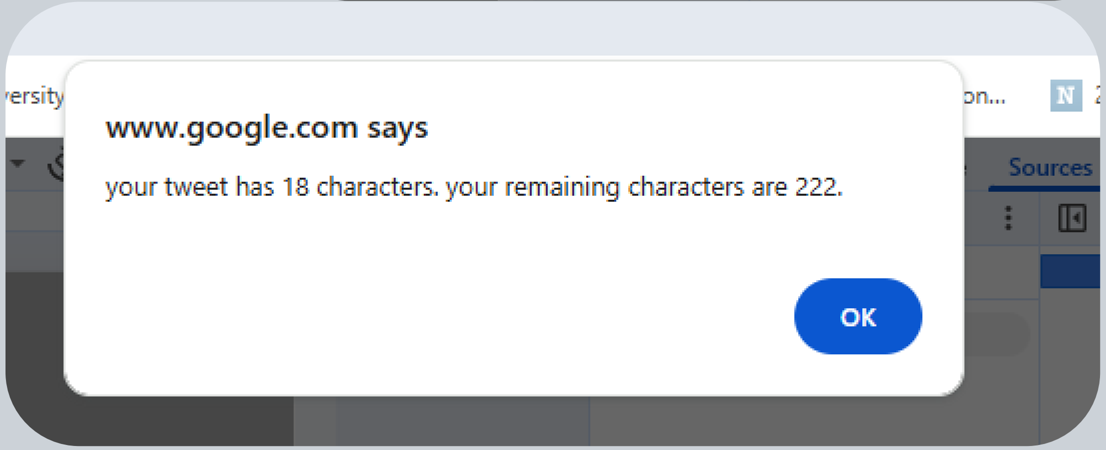
Write your tweet



www.google.com says

Write your tweet

This is my tweet.



www.google.com says

your tweet has 18 characters. your remaining characters are 222.

Slicing in Strings

- It is used to extract or sliceout the specific range of letters

```
var name="Pakistan";
name.slice(0,1);
```

```
P
```

```
var name="Pakistan";
name.slice(0,3);
```

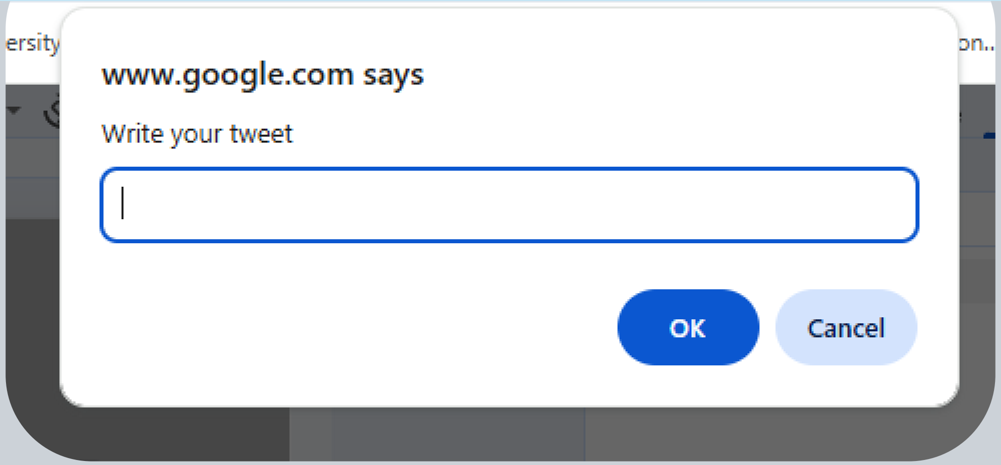
```
Pak
```

```
var name=
"Computer";name.slice(2,6);
```

```
mput
```

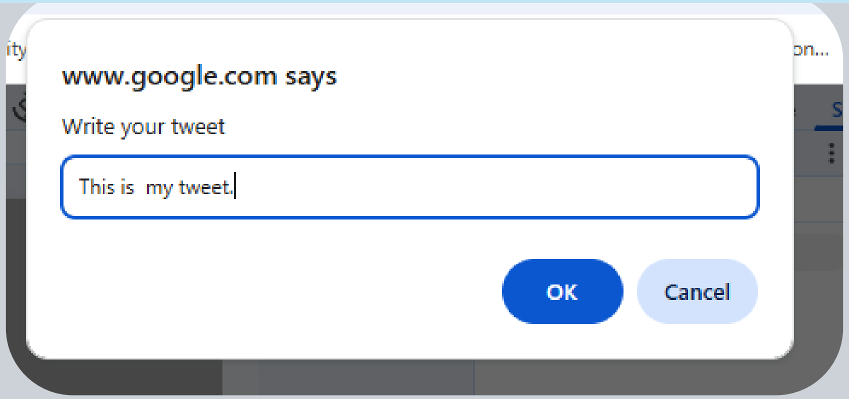
Tweet Web App 2.0

```
var tweet = prompt ("Write your tweet");
alert ("your tweet has "+tweet.length+" characters. your remaining characters are "+(10-tweet.length)+".");
alert("Your tweet is "+tweet.slice(0,10));
```



www.google.com says

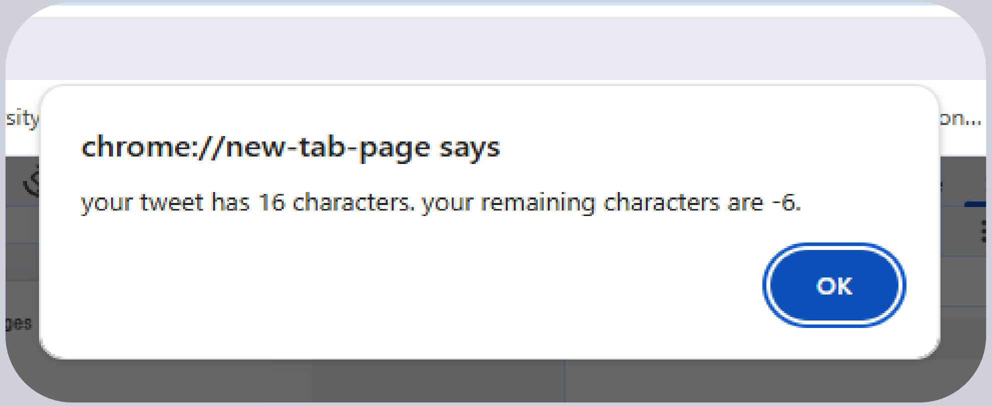
Write your tweet



www.google.com says

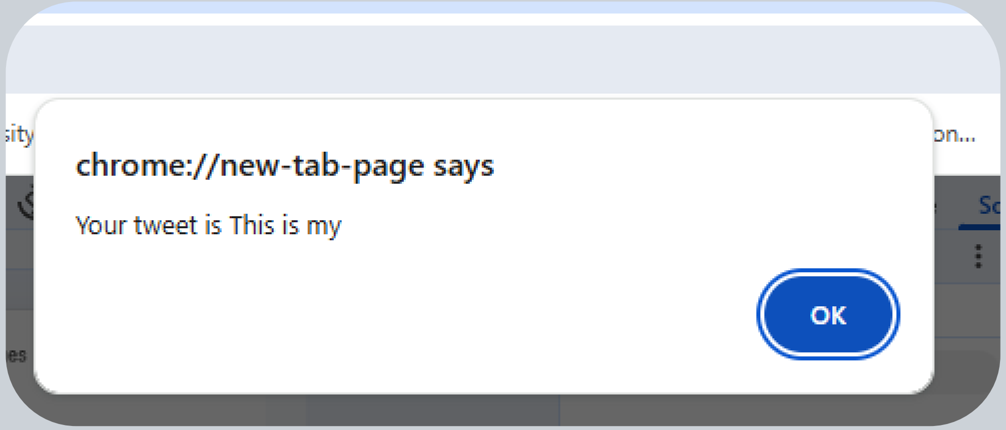
Write your tweet

This is my tweet.



chrome://new-tab-page says

your tweet has 16 characters. your remaining characters are -6.

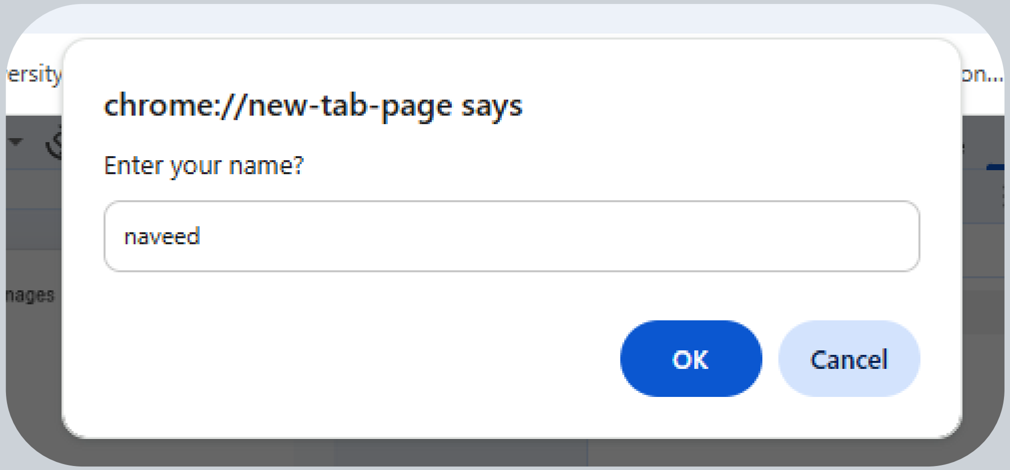


chrome://new-tab-page says

Your tweet is This is my

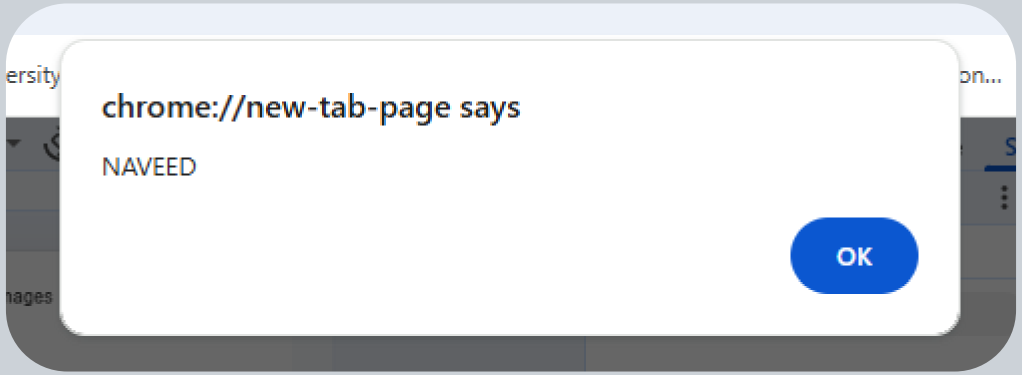
toUpperCase()

```
var name=prompt("Enter your name?");
alert(name.toUpperCase());
```


chrome://new-tab-page says

Enter your name?

naveed


chrome://new-tab-page says

NAVEED

Practice Problem

- Write a JavaScript program that asks the user to **"Enter your name"**, and then displays it back using alert(), ensuring that **only the first letter is capitalized**.

- Solution:

- var name=prompt("Enter your name?");
- alert(name.slice(0,1).toUpperCase()+name.slice(1,name.length).toLowerCase());

Arithmetic in JavaScript

- .

```
var a=2+2; //addition
var b=10-6; // substraction
var c=3*3; // multiplication
var d=5/2; // division
var e=7%3; // modulo operator
```

```
a=4
b=4
c=9
d=2.5
e=1
```

Increment/ Decrement in JavaScript

```
Var a=1;                //a=1
a++;                    // a=2
a+=2;                   // a=4
a-=3;                   // a=1
a--;                    //a=0
var b=3;                //b=3
a+=b;                   //a=3
```

- These operations are also valid with *,- and / .

Functions

- Creating a function

```
function function_name() { }
```

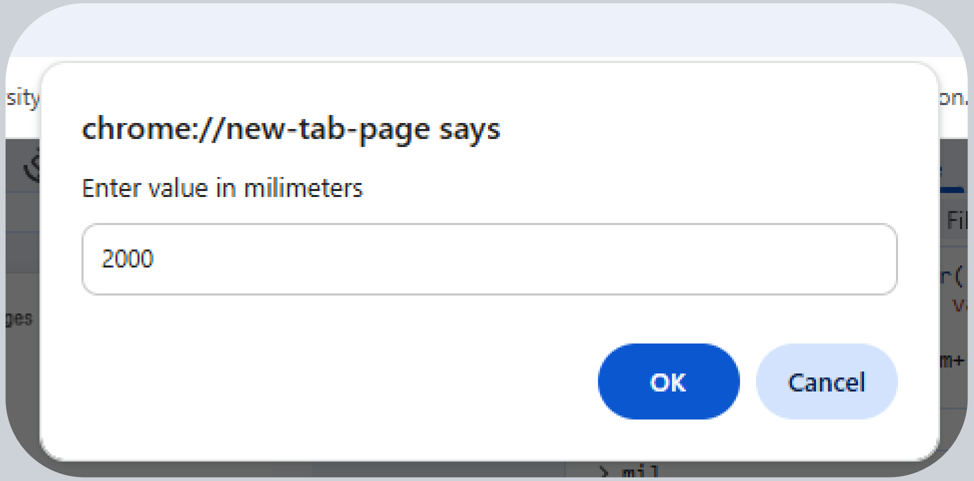
- Calling function

```
function_name();
```

Functions - Example

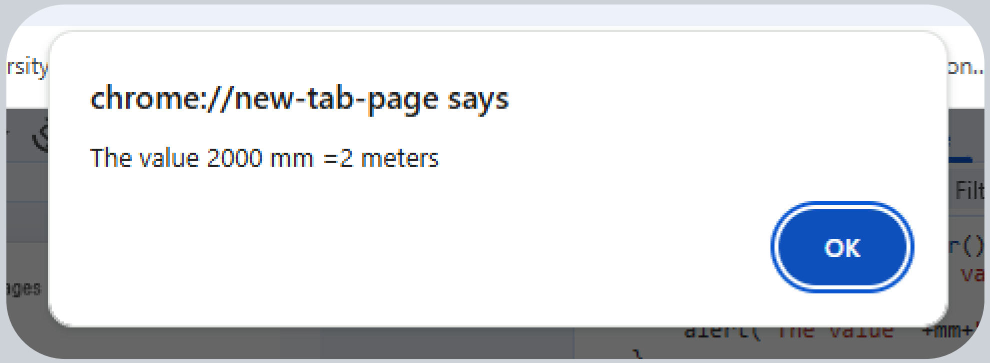
-
  ```
  function milimeterToMeter(){
          var mm=prompt("Enter value in milimeters");
          var meter=mm/1000;
          alert("The value "+mm+" mm ="+meter+"meters");
  }
  ```

```
milimeterToMeter();
```


chrome://new-tab-page says

Enter value in milimeters

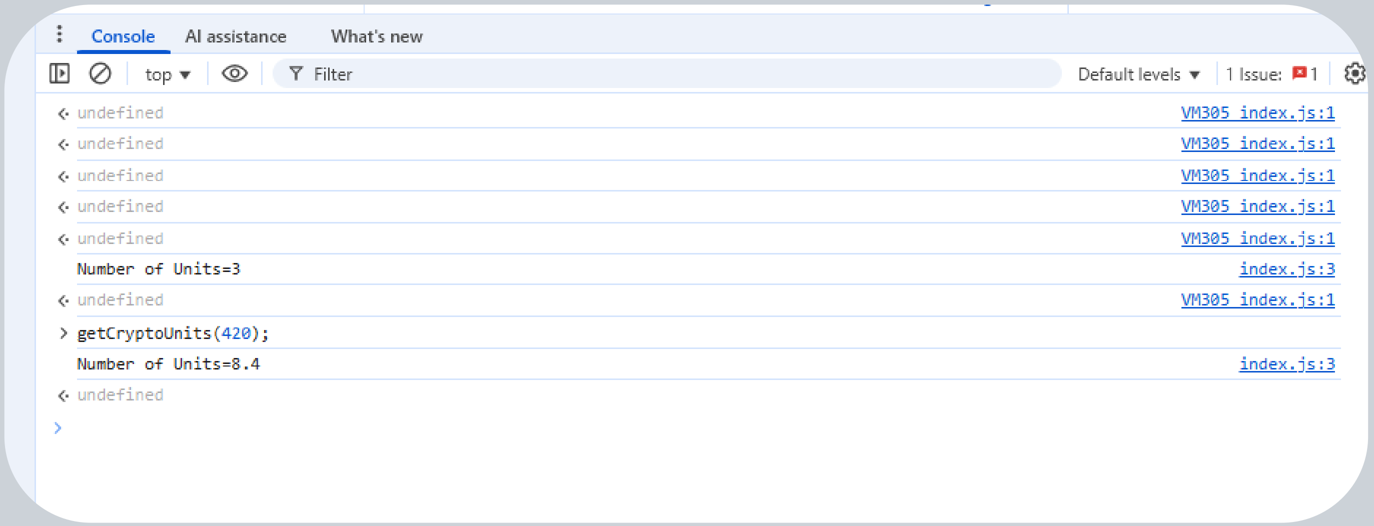
2000


chrome://new-tab-page says

The value 2000 mm =2 meters

Functions - Argument

-
  ```
  function getCryptoUnits(money){
      var pricePerUnit=50;
      console.log("NumberUnits="+money/pricePerUnit);
  }
  getCryptoUnits(150);
  ```

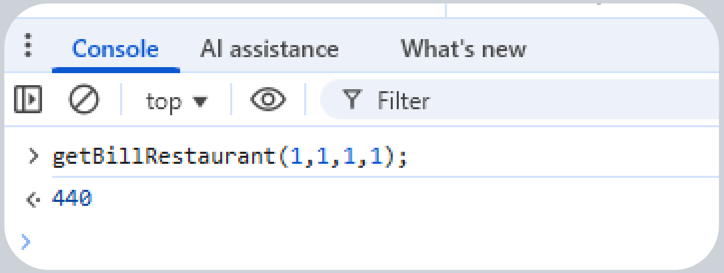


```
<· undefined                                    VM305 index.js:1
<· undefined                                    VM305 index.js:1
<· undefined                                    VM305 index.js:1
<· undefined                                    VM305 index.js:1
<· undefined                                    VM305 index.js:1
   Number of Units=3                                  index.js:3
<· undefined                                    VM305 index.js:1
>  getCryptoUnits(420);
   Number of Units=8.4                                index.js:3
<· undefined
>
```

Functions - Argument & Return

```
function getBillRestaurant(briyaniPlates,numColdDrinks,numSalad,numTeaCups){
        var pricePerBriyaniPlate=250;
        var priceColdDrink =70;
        var priceSalad=50;
        var priceTeaCup=70;
        var costBriyani=pricePerBriyaniPlate*briyaniPlates;
        var costColdDrinks=priceColdDrink*numColdDrinks;
        var costTea=priceTeaCup*numTeaCups;
        var costSalad=numSalad*priceSalad;
        var bill= costBriyani+costColdDrinks+costTea+costSalad;
        return bill;
}
```

```
getBillRestaurant(1,1,1,1);
```



```
> getBillRestaurant(1,1,1,1);
<· 440
>
```

Random Number Generation

- Var n=Math.random();
  - It generates the random number between:
    - 0 and 0.999...(16 Decimal Places or 16 Significant digits)

```
function rollTheDice(){
        //return Math.ceil(Math.random()*6);
        var n=Math.random();
        n=n*6;
        n=Math.ceil(n);
        return n;
}

rollTheDice();
```

If else Structure – Control flow Instructions

- .

```
function toCheckGpa(score){
        if (score===4) {
                console.log("GPA is 4");
        }
        else{
                console.log("GPA is not 4");
        }
}
```

JavaScript Comparison Operators (Equality & Relational)

- .

Operator	Meaning / Description	Example	Result
==	Equal to (compares values, ignores type)	5 == "5"	true
===	Strict equal to (compares value **and** type)	5 === "5"	false
!=	Not equal to (compares values, ignores type)	5 != "5"	false
!==	Strict not equal to (compares value **and** type)	5 !== "5"	true
>	Greater than	8 > 3	true
<	Less than	2 < 5	true
>=	Greater than or equal to	7 >= 7	true
<=	Less than or equal to	4 <= 6	true

Difference Between == and === in JavaScript

- .

Example	With == (Loose Equality)	With === (Strict Equality)	Explanation
"5" == 5	✅ true	❌ false	== converts "5" to number → 5 == 5 → true; === checks type → string ≠ number
0 == false	✅ true	❌ false	false becomes 0 → 0 == 0 → true; === compares number vs Boolean
null == undefined	✅ true	❌ false	== treats both as "empty" values; === sees them as different types
"0" == false	✅ true	❌ false	"0" → 0, and false → 0; equal after conversion
1 == true	✅ true	❌ false	true → 1 → 1 == 1 → true; but not same type
0 == "0"	✅ true	❌ false	"0" converts to 0; but different types
5 == 5	✅ true	✅ true	Both same value and type (number)

Difference Between == and === in JavaScript

-
  -

Operator	Type Conversion	Recommended?	Use When
==	✔️ Performs automatic type conversion	⚠️ Not recommended	You are sure type conversion won’t cause errors
===	❌ No type conversion	✅ Recommended	You want accurate, predictable comparisons

Combining Comparaters

Operator	Name / Meaning	Example	Result	Description
&&	Logical AND	(x > 5 && y < 10)	true if both are true	Returns true only if **both** conditions are true
` ||	Logical OR	(x > || y<10)	True if any option is true	Returns true only if any of the condition is true
!	Logical NOT	!(x > 5)	Opposite of condition	Reverses the boolean value (true → false, false → true)

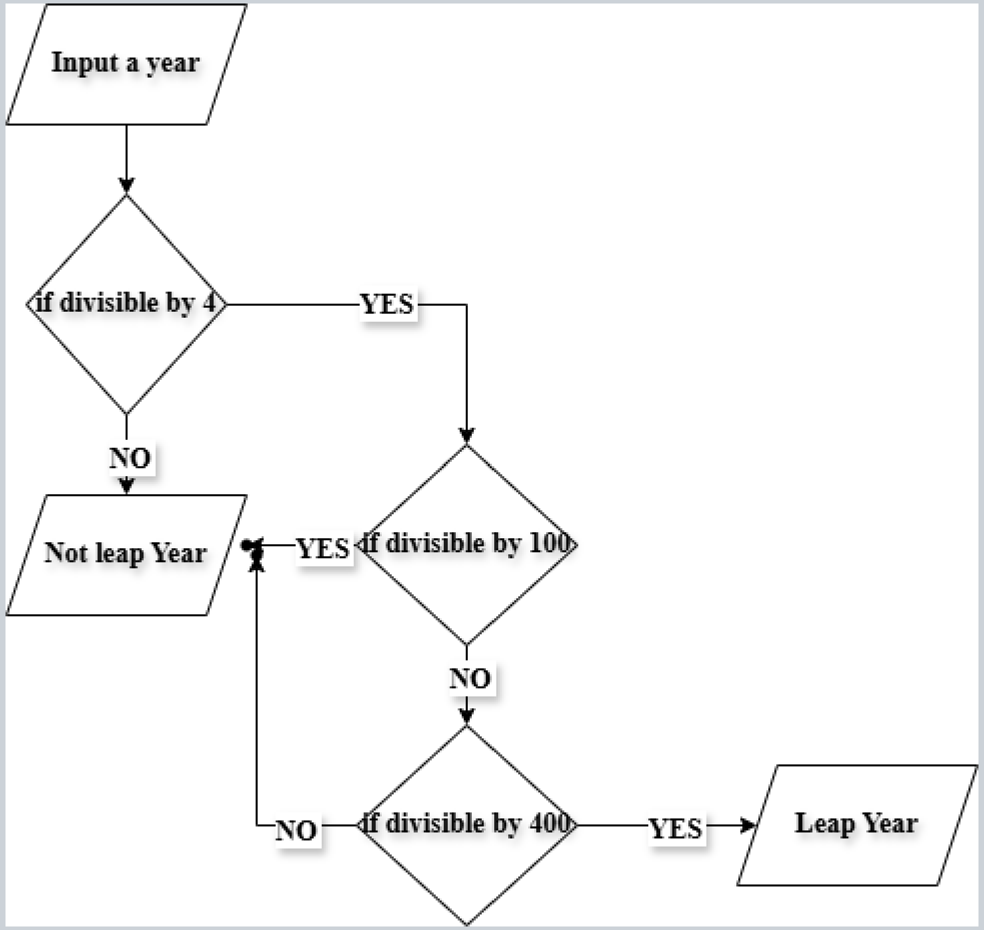
Combining Comparators

-

```
function bmiCalculator (weight, height) {
        var bmi=weight/(height*height);
        var interpretation;
         if(bmi<18.5){      interpretation="Your BMI is "+bmi+", so you are underweight.";
                }
        else if(bmi >=18.5 && bmi<=24.9) {
                interpretation="Your BMI is "+bmi+", so you have a normal weight.";
                } else {
                        interpretation="Your BMI is "+bmi+", so you are overweight.";
                        }
        return interpretation;
}
```

Leap Year Condition

- **Logic for Leap Year**
  - A year is a leap year if:
  - It is divisible by **4,**
  - **But not** divisible by **100,**
  - **Unless** it is also divisible by **400**.

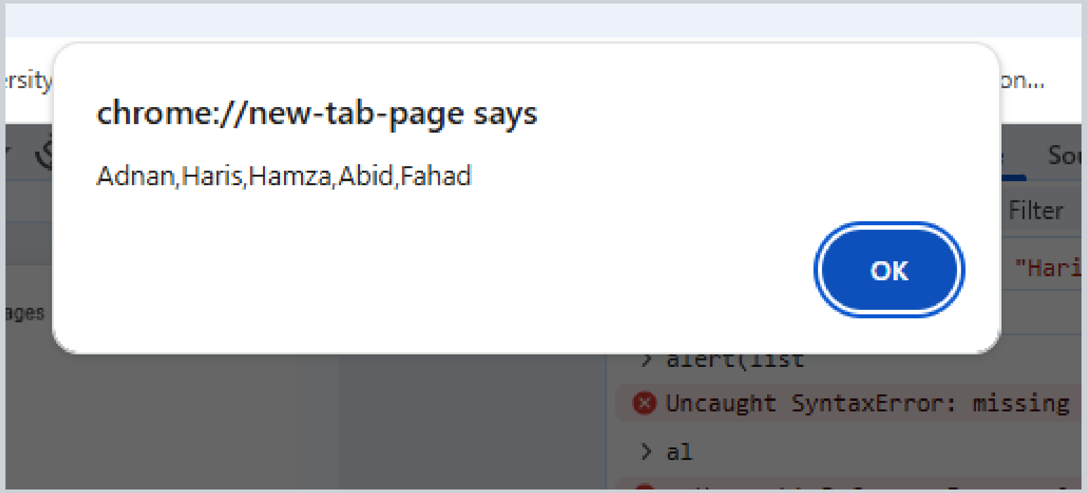


Leap year Prediction

```
function isLeap(year) {
        var result;
                if (year%4 === 0){
                        result="Leap year";
                }
                        else if(year%100 === 0){
                                if(yar % 400 === 0){
                                        result="Leap year";
                                } else {
                                        result="Not leap year";
                                        }
                                }
                                else result="Not leap year";
                                return result;}
```

Arrays

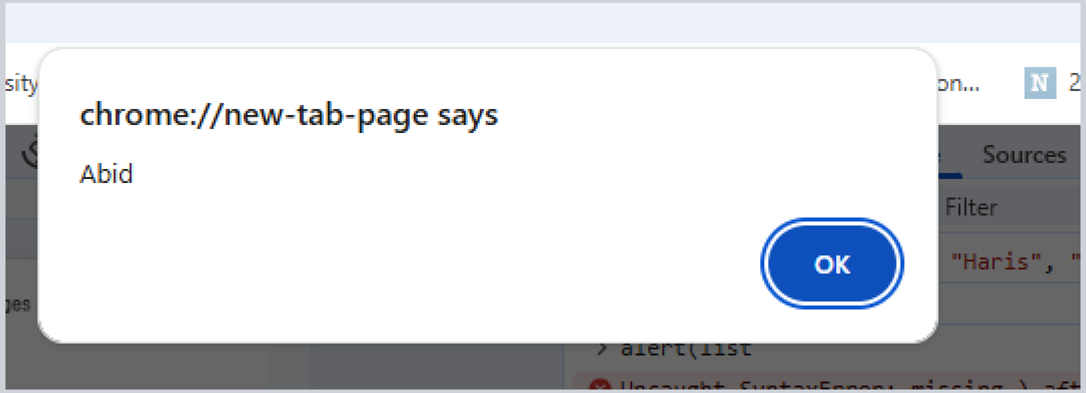
- var list_names=["Adnan", "Haris", "Hamza", "Abid", "Fahad"];
- alert(list_names);



chrome://new-tab-page says

Adnan,Haris,Hamza,Abid,Fahad

- alert(list_names[3]);



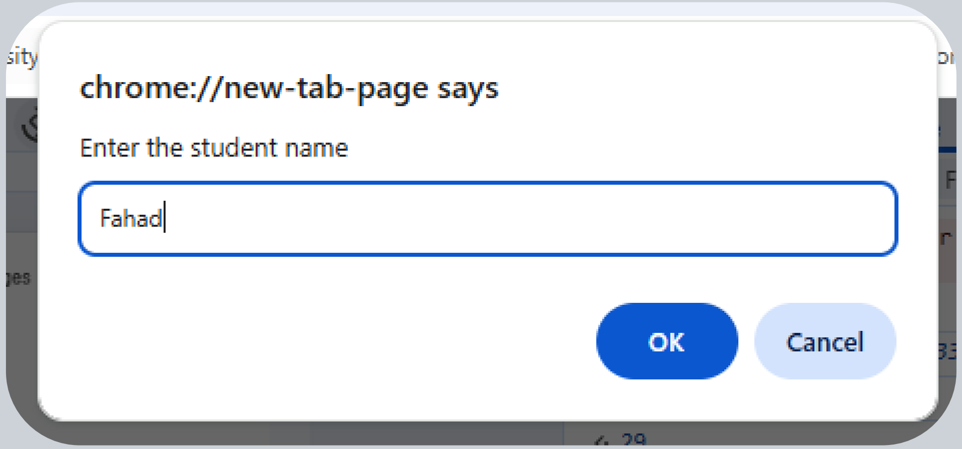
chrome://new-tab-page says

Abid

Arrays (.includes function)

- .

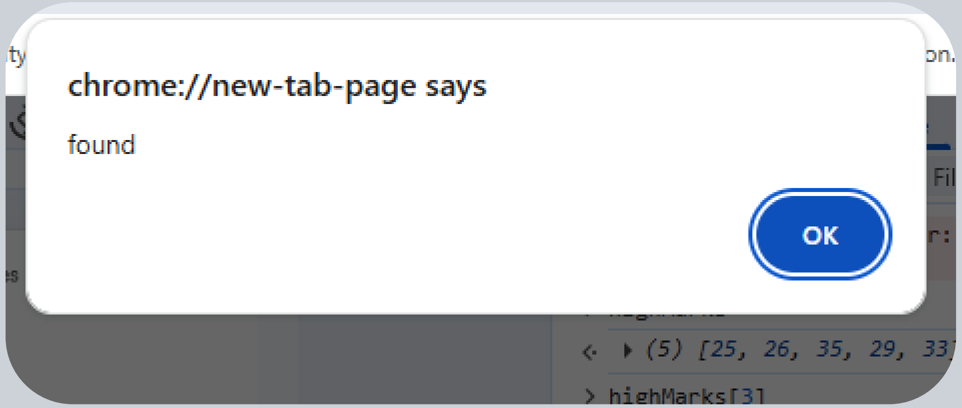
```
var list_names=["Adnan", "Haris", "Hamza", "Abid", "Fahad"];

var guestName=prompt("Enter the student name");
if( list_names.includes(guestName)) {
            alert ("found");
            }
            else
                        {
                        alert("not found");
            }
```



chrome://new-tab-page says

Enter the student name

Fahad

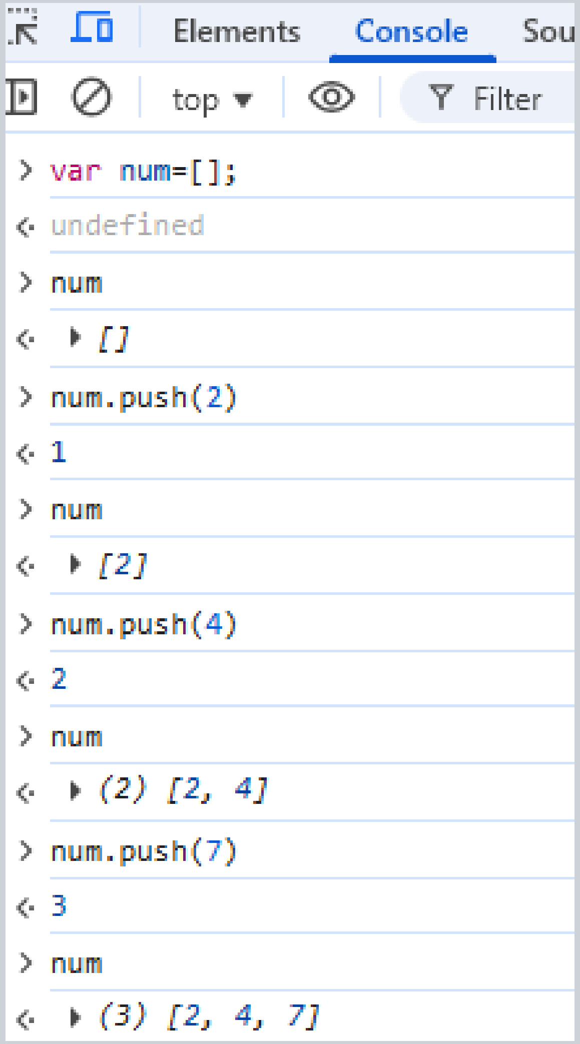


chrome://new-tab-page says

found

array .push() and .pop() functions

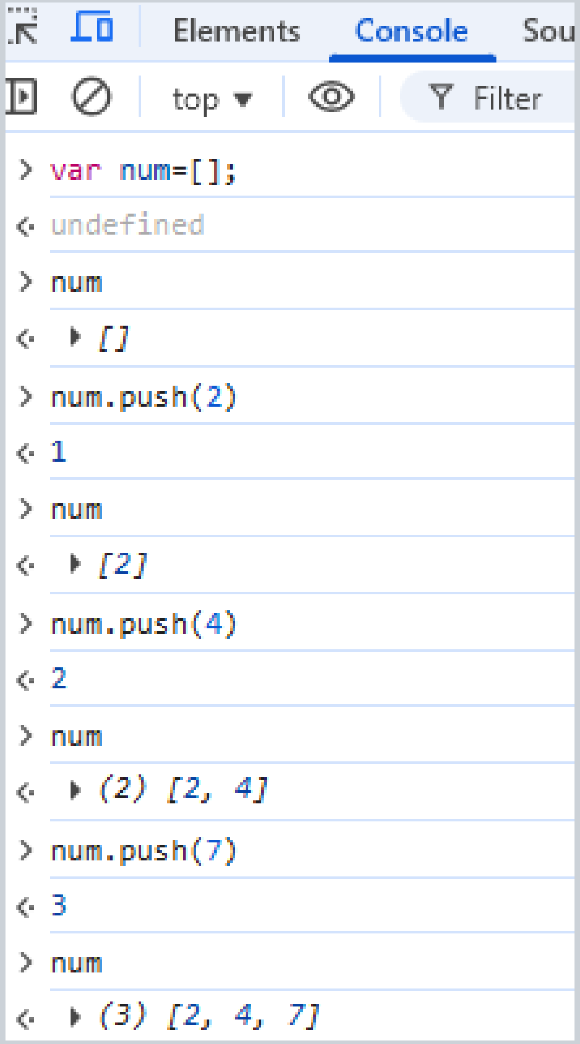
- .



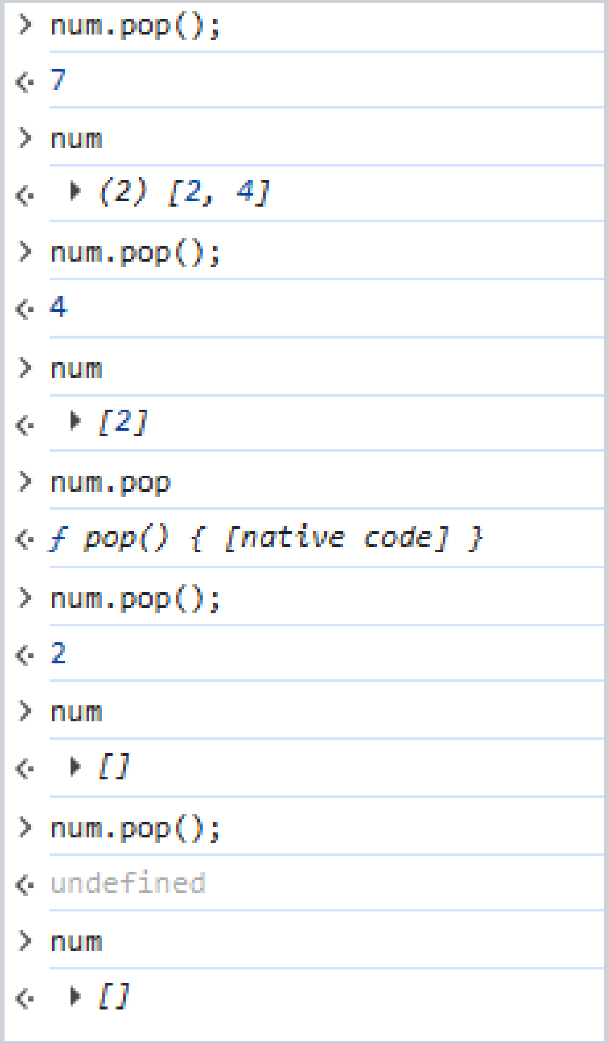
```
> var num=[];
<· undefined
> num
<·  ▸ []
> num.push(2)
<· 1
> num
<·  ▸ [2]
> num.push(4)
<· 2
> num
<·  ▸ (2) [2, 4]
> num.push(7)
<· 3
> num
<·  ▸ (3) [2, 4, 7]
```

array .push() and .pop() functions

- .



```
> var num=[];
<· undefined
> num
<· ▸ []
> num.push(2)
<· 1
> num
<· ▸ [2]
> num.push(4)
<· 2
> num
<· ▸ (2) [2, 4]
> num.push(7)
<· 3
> num
<· ▸ (3) [2, 4, 7]
```



```
> num.pop();
<· 7
> num
<· ▸ (2) [2, 4]
> num.pop();
<· 4
> num
<· ▸ [2]
> num.pop
<· ƒ pop() { [native code] }
> num.pop();
<· 2
> num
<· ▸ []
> num.pop();
<· undefined
> num
<· ▸ []
```

fizzBuzz() – problem statement

- Write a JavaScript function named **fizzBuzz()** that, when called, builds an array containing the numbers from **1 to 100**, but follows these rules:
  - Replace numbers divisible by **3** with "Fizz".
  - Replace numbers divisible by **5** with "Buzz".
  - Replace numbers divisible by **both 3 and 5** with "FizzBuzz".
  - Each time the function runs, it should update and display the array.

fizzBuzz() – problem statement

- .

```
var output=[];
var i=1;

function fizzBuzz(){
        if (i % 15===0)
                output.push("FizzBuzz");
        else if(i % 5 ===0){
        output.push("Buzz");
        }
        else if(i%3===0){
        output.push("Fizz");
        } else {
                output.push(i);
                }
        i++;
        console.log(output);
        }
```

JavaScript for Loop

- **Syntax:**

```
for (initialization; condition; increment/decrement) {
   // code to be executed
}
```

Explanation:

- **Initialization:** Runs once before the loop starts.
- **Condition:** Checked before each iteration. If false → loop ends.
- **Increment/Decrement:** Updates the loop variable after each iteration.

```
for (var i = 1; i <= 5; i++) {
   console.log("Count: " + i);
}
```

JavaScript while Loop

- **Syntax:**

```
while (condition) {
   // code to be executed
}
```

Explanation:

- The loop runs **as long as the condition is true.**
- You must update the variable **inside** the loop to avoid infinite loops.

Example:

```
var i = 1;
while (  i <= 5) {
   console.log("Number: " + i);
   i++;
}
```