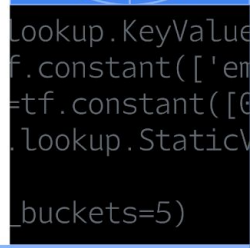
Google Developer Groups

Session starting soon

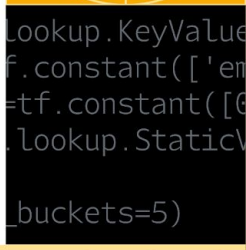


Google Developer Groups

Welcome To Day <3/>

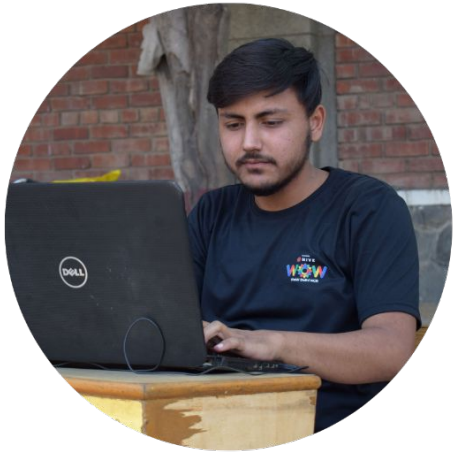
<TECH WINTER BREAK/>

Level up your skills during winter chillllsss



Google Developer Groups

About the Mentor



Sagar Kumar Jha is an accomplished professional with diverse experience across **high-impact organizations.**

- Interned at **DRDO**, focusing on **APK & API Security**.
- Interned at the **Special Protection Groups(SPG)**, specializing in **Radar Communication Systems**.
- Interned at **NITI Aayog**, contributing to **Software Engineering projects**.

Google Developer Groups

About the Instructor

Divyansh Raj is a skilled professional currently serving as the Technical Lead of GDGC and an intern at Delhi Government University, showcasing his expertise in technical leadership and project execution.

Google Developer Groups

Connect With Us !

Sagar Kumar Jha
(Mentor)

Divyansh Raj
(Instructor)

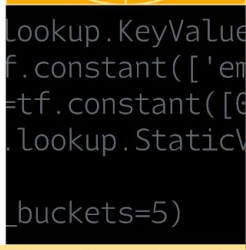
```
lookup.KeyValue
f.constant(['em
=tf.constant([0
.lookup.StaticV

_buckets=5)
```

Google Developer Groups

Day 3: Javascript

The Key to Dynamic Web Development !



<TECH WINTER BREAK/>

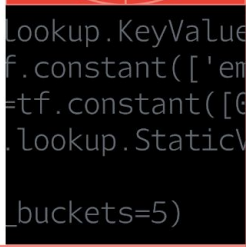
Level up your skills during winter chillllsss

Google Developer Groups

Let's Start!

Javascript: Used to make websites dynamic.

Javascript is the logic of the web and the only programming language which the browser understands.



1. Introduction to JavaScript

What is JavaScript?

A versatile, high-level programming language.
Enables dynamic behaviour on websites.
Runs on the client-side and server-side (e.g., Node.js).

Key Features:

Lightweight and interpreted.
Prototype-based, object-oriented.
Event-driven and asynchronous capabilities.

```
console.log('Hello, JavaScript!');
```

2. Javascript Basics

Operators

Types of Operators:
1. Arithmetic: +, -, *, /, %.
2. Comparison: ==, ===, !=, !==, >, <, >=, <=.
3. Logical: &&, ||, !
4. Assignment: =, +=, -=.
5. Bitwise: &, |, ^, ~, <<, >>.
6. Type: typeof, instance of

3. Conditional Statements

Types of Conditions:
1. If, Else, Else if
2. Switch

Best Practices:
- Use === for strict equality checks.
- Use default in switch to handle edge cases.

4. Loops

Types of Loops:
1. For
2. While
3. Do....while

```
for (let i = 1; i <= 5; i++) {
    console.log(i);
}

let i = 1;
while (i <= 5) {
    console.log(i);
    i++;
}

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

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

5 . Functions

What are functions?

A block of code designed to perform a task.

Types:

Function Declaration
Function Expression
Arrow Function
Immediately Invoked Function Expressions

Best Practices:

Use descriptive names for functions.
Keep functions focused on a single task.

```
function greet(name) {
    return `Hello, ${name}!`;
}

const add = (a, b) => a + b;

(function sayHi() {
    console.log('Hi there!');
})(); // IIFE

console.log(greet('Alice')); // Hello, Alice!
console.log(add(2, 3)); // 5
```

6 . Objects

What are Objects?

Collection of Key-Value Pairs.

Accessing Properties:

Dot Notation: object.property

Bracket Notation: object[‘property’]

```
const car = {
    brand: 'Toyota',
    model: 'Corolla'
};

car.year = 2020; // Adding
console.log(car);
delete car.model; // Deleting
console.log(car);
```

7. Arrays

What are Arrays?

Ordered Collection of data

Common Methods:

Mutating: push, pop, shift, unshift, splice.

Non-mutating: slice, concat, map, filter, reduce.

```
const fruits = ['apple', 'banana', 'cherry'];
fruits.push('date');
console.log(fruits); // ['apple', 'banana', 'cherry', 'date']

const numbers = [1, 2, 3, 4, 5];
const doubled = numbers.map(num => num * 2);
console.log(doubled); // [2, 4, 6, 8, 10]
```

8. More about Functions

1. Promises
2. Async & Await

https://medium.com/@hassanahmedkhan/a-beginners-guide-to-promise-api-in-javascript-c8a2fe365cdc

Asynchronous Programming

JavaScript is single-threaded and asynchronous programming is used to handle operations like API calls, timers, or file reading without blocking the main thread.

What is a Callback?

A callback is a function passed as an argument to another function, which gets executed after the completion of an operation. **Callbacks are often used for handling asynchronous tasks.**

When multiple asynchronous tasks depend on each other, nesting callbacks can make the code difficult to read, debug, and maintain. This is known as **callback hell.**

```
function fetchData(callback) {
  setTimeout(() => {
    console.log("Data fetched");
    callback(); // Execute the callback function
  }, 1000); // Simulate an asynchronous task with a delay of 1 second
}

// Call the fetchData function and pass a callback
fetchData(() => {
  console.log("Processing data...");
});
```

Promises

Promise as an object that represents the eventual completion (or failure) of an asynchronous operation.

Explain the three states of a promise:

- Pending: Initial state.
- Fulfilled: Operation completed successfully.
- Rejected: Operation failed.

Chained Promises

```
// How multiple promises can be chained together
fetchData
  .then((data) => {
    console.log(data);
    return "Processing data";
  })
  .then((result) => console.log(result))
  .catch((error) => console.error(error));
```

Async & Await

async functions return promises and await pauses execution until the promise resolves.

```
async function fetchDataAsync() {
  try {
    const data = await new Promise((resolve, reject) =>
      setTimeout(() => resolve("Data fetched"), 1000)
    );
    console.log(data);
  } catch (error) {
    console.error(error);
  }
}

fetchDataAsync();
```

9. API Fetching

An API (Application Programming Interface) is a set of rules, protocols, and tools that allows different software applications to communicate with each other.

It acts as a bridge that enables a client (like a frontend application) to interact with a server or another application.

```
async function fetchQuote() {
  try {
    const response = await fetch("https://api.kanye.rest/");
    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }
    const data = await response.json();
    console.log(`Kanye Quote: "${data.quote}"`);
  } catch (error) {
    console.error("Error fetching quote:", error);
  }
}
```

10. Document Object Model

What is DOM?

The Document Object Model represents the HTML structure as objects.

Selecting Elements:

By ID: document.getElementById(‘id’)
By Class: document.getElementByClassName(‘class’)
By Tag: document.getElementByTagName(‘tag’)
CSS Selectors: document.querySelector(‘selector’)
Multiple Selectors: document.querySelectorAll(‘selector’)

```
const heading = document.querySelector('h1');
heading.textContent = 'Welcome to JavaScript!';
```

Google Developer Groups

Thank You!

That’s all for today!

Happy New Year everyone,

The fun is in the journey, success is just a place to be at.