



Session starting soon

```
Lookup.KeyValue  
f.constant(['en  
=tf.constant([G  
.lookup.StaticV  
_buckets=5)
```



Welcome To Day <4/>

<TECH WINTER BREAK/>

Level up your skills during winter chillllssss

```
Lookup.KeyValue  
f.constant(['en  
=tf.constant([  
.lookup.StaticV  
_buckets=5)
```



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**.



```
Lookup.KeyValue  
F.constant(['em  
=tf.constant([0  
.lookup.StaticV  
_buckets=5)
```



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.



```
Lookup.KeyValue  
f.constant(['em  
=tf.constant([0  
.lookup.StaticV  
_buckets=5)
```



Connect With Us !



**Sagar Kumar Jha
(Mentor)**



**Divyansh Raj
(Instructor)**

```
Lookup.KeyValue
f.constant(['en
=tensor([G
.lookup.StaticV
_buckets=5)
```



Day 4: DOM Manipulation

Document Object Model

<TECH WINTER BREAK/>

Level up your skills during winter chillllssss



Let's Start!

DOM Manipulation:

programming interface for web documents.

Represents the structure of an HTML or XML document as a tree of objects.
Enables dynamic updates to content, structure, and styles.

1. Introduction to DOM Manipulation

Accessing the HTML of the project through the javascript.

Key Features:

Enables interactive and dynamic web pages.

Core concept for modern JavaScript frameworks and libraries

- Tree:
 - html
 - body
 - h1
 - p

```
<html>
  <body>
    <h1>Title</h1>
    <p>Hello, World!</p>
  </body>
</html>
```

2. Accessing the DOM

Methods:

- document.getElementById(id)
- document.getElementsByClassName(className)
- document.getElementsByTagName(tagName)
- document.querySelector(selector)
- document.querySelectorAll(selector)

Example:

```
const header = document.getElementById('header');
const paragraphs = document.querySelectorAll('p');
```

3. Manipulating Elements

Changing Content

- element.innerHTML
- element.textContent

Changing Attributes

- element.setAttribute(attribute, value)
- element.getAttribute(attribute)
- element.removeAttribute(attribute)

Example:

```
const title = document.querySelector('h1');
title.textContent = 'New Title';
const link = document.querySelector('a');
link.setAttribute('href', 'https://example.com');
```

4. Manipulating Styles

Inline Styles:

- element.style.property = value;

CSS Classes:

- element.classList.add(className)
- element.classList.remove(className)
- element.classList.toggle(className)
- element.classList.contains(className)

Example:

```
const box = document.querySelector('.box');
box.style.backgroundColor = 'blue';
box.classList.add('highlight');
```

5. Creating and Removing Elements

Creating Elements:

- `document.createElement(tagName)`
- `document.createTextNode(text)`

Appending Elements:

- `parentNode.appendChild(newNode)`
- `parentNode.insertBefore(newNode, referenceNode)`

Removing Elements:

- `parentNode.removeChild(childNode)`

Example:

```
const newDiv = document.createElement('div');
newDiv.textContent = 'Hello, DOM!';
document.body.appendChild(newDiv);
```

6 . Event Handling

Adding Events:

- element.addEventListener(event, handler)
- Inline Events (not recommended)

Removing Events:

- element.removeEventListener(event, handler)

Event Types:

- click, mouseover, keydown, etc.

```
const button = document.querySelector('button');
button.addEventListener('click', () => alert('Button clicked!'))
```

7. Traversing the DOM

Navigating Nodes:

- parentNode, childNodes, firstChild, lastChild
- nextSibling, previousSibling

Navigating Elements:

- parentElement, children, firstElementChild, lastElementChild
- nextElementSibling, previousElementSibling

Example:

```
const list = document.querySelector('ul');
console.log(list.firstElementChild.textContent);
```

8. Real Examples

Dynamic To-Do List:

- Add, edit and delete tasks.

Form Validation:

- Real-time input validation

Interactive Image Gallery:

- Thumbnail clicks to view larger images.

9. Problems with DOM Manipulation

Performance Issues:

- Frequent reflows and repaints due to DOM updates.
- Inefficient queries and traversal.

Complexity in Large Applications:

- Managing state across multiple DOM nodes.
- Hard to debug code when updates are not centralized.

Cross Browser Inconsistencies:

- Variations in DOM method support and behaviour.

Security Concerns

- Risk of XSS attacks when using innerHTML with untrusted input

10. Frameworks and Libraries

Why Use Frameworks?

- Simplify and abstract DOM manipulation.
- Provide tools for state management and rendering.

Popular Frameworks:

- React: Virtual DOM for optimized rendering.
- Vue.js: Easy-to-learn, reactive programming model.
- Angular: Complete framework with a focus on two-way data binding.

Advantages:

- Better performance (Virtual DOM).
- Cleaner and more maintainable code.
- Ecosystem and community support



Thank You!

That's all for today!

```
Lookup.KeyValue  
f.constant(['en  
=tf.constant([G  
.lookup.StaticV  
_buckets=5)
```